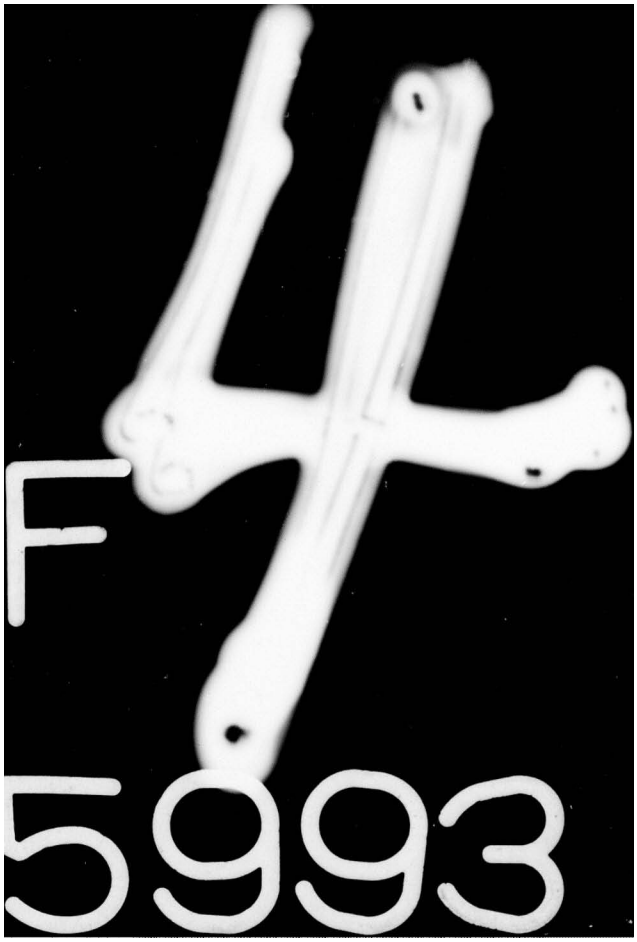
AD-A035 993 PRC INFORMATION SCIENCES CO MCLEAN VA F/G 8/2
ACS SYMBOLIZATION FOR DMAAC. VOLUME II. COMPUTER PROGRAM DOCUME--ETC(U)
NOV 76 P D BELL, J A NEUFFER, M L TAYLOR F30602-75-C-0319
UNCLASSIFIED RADC-TR-76-334-VOL-2 NL

1 OF 4
ADA035993


F
5993

AD A035993

(12)

RADC-TR-76-334, Volume II (of two)
Final Technical Report
November 1976

# ACS SYMBOLIZATION FOR DMAAC
## Computer Program Documentation

PRC Information Sciences Company

Approved for public release;
distribution unlimited.

DDC
RECEIVED
FEB 24 1977
A

ROME AIR DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
GRIFFISS AIR FORCE BASE, NEW YORK 13441

COPY AVAILABLE TO DDC DOES NOT
PERMIT FULLY LEGIBLE PRODUCTION

This report contains a large percentage of machine-produced copy which is not of the highest printing quality but because of economical consideration, it was determined in the best interest of the government that they be used in this publication.

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

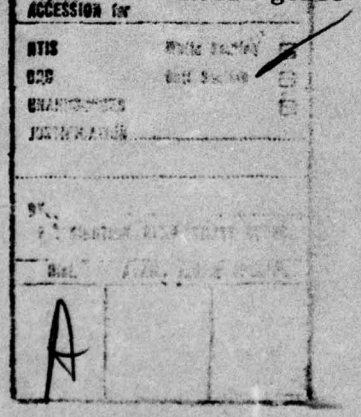
This report has been reviewed and approved for publication.

APPROVED: John R Baumann

JOHN R. BAUMANN
Project Engineer

APPROVED: Howard Davis

HOWARD DAVIS
Technical Director
Intelligence & Reconnaissance Division




FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| (19) REPORT DOCUMENTATION PAGE | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|
| (18) 1. REPORT NUMBER: RADC-TR-76-334, Vol II (of two) 2. | 2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER |
| (6) 4. TITLE (and Subtitle): ACS SYMBOLIZATION FOR DMAAC. Volume II. Computer Program Documentation. | (9) 5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report. June 1975 - April 1976 |
| | 6. PERFORMING ORG. REPORT NUMBER: N/A |
| (10) 7. AUTHOR(s): Paul D. Bell, John A. Neuffer, M. Lynn Taylor | (15) 8. CONTRACT OR GRANT NUMBER(s): F30602-75-C-0319 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: PRC Information Sciences Company, 7600 Old Springhouse Road, McLean VA 22101 | (16) 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 64701B, 32020316 (17) 03 |
| (11) 11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (IRRP), Griffiss AFB NY 13441 | 12. REPORT DATE: November 1976 |
| | 13. NUMBER OF PAGES: 328 |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same (12) 342p. | 15. SECURITY CLASS. (of this report): UNCLASSIFIED |
| | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer:
John R. Baumann (IRRP)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Cartography
Symbolization
Line Smoothing
Cartography

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

RADC has implemented a Graphic Line Symbolization System (GLSS) on the Univac 1108 computer system located at the Defense Mapping Agency Aerospace Center (DMAAC).

The software accepts data in the DMAAC Lineal Input System format and creates symbolized line data for final color separation plotting. Symbols are applied according to Joint Operations Graphics (JOG) specifications.

(see reverse)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

LB

407126

In addition to the lineal symbol capability, the GLSS is capable of creating a significant number of point symbols.

The software is written in ASCII COBOL and Fortran V languages and requires approximately 40K words of memory for loading an execution of all functions.

The software configuration is highly segmented into areas of job set-up, file input, job monitoring, symbol application control, symbol specification correlation, symbol application processes, line smoothing and data culling, job reporting, and file output.

# PREFACE

This is Volume II of a two volume final technical report prepared by PRC Information Sciences Company, 7600 Old Springhouse Road, McLean Virginia. The report covers work performed under Contract F30602-75-C-0319 for Rome Air Development Center, Griffiss Air Force Base, New York. The report describes work performed from June 1975 through April 1976. Mr. John R. Baumann (IRRC) was the RADC Project Engineer, Mr. Frank Mirkay was the DMAAC technical coordinator, and Mr. M. Lynn Taylor was the PRC Project Manager.

## ABSTRACT

PRC/ISC, under a RADC contract, converted and expanded the Graphic Line Symbolization System (GLSS) to operate in the production environment at the Defense Mapping Agency Aerospace Center (DMAAC). The system was converted from the HIS-635 to the UNIVAC 1108. Major enhancements to the original software system included additional point symbol capabilities and additional output file formats for interfacing with DMAAC plotter systems. Major characteristics of GLSS includes:

- o Hardware - UNIVAC 1108
- o Software - written in COBOL and FORTRAN V compiler languages and operates under EXEC 8.
- o Modularity - the software configuration is highly segmented into areas of job setup, file input, job monitoring, symbol application control, symbol specification correlation, symbol application processes, line smoothing, job reporting, and file output.
- o Resources - requires approximately 40K words of memory for loading and execution of all software; program overlaying or selective loading of required software can significantly reduce core storage.

GLSS provides a wide range of data processing capabilities related to cartographic symbology. Various capabilities, options and processing techniques of GLSS includes the following:

Dashing

- o Variable sizes of dashes and spaces
- o Feature must start and end with dash at least $\frac{1}{2}$ length of dash size
- o Dash must carry through points flagged as special points

Casing

- o Variable size cases
- o Line quality of case should equal or exceed quality of original line center

Ticking

- Full tick
- Alternating half tick
- Half tick (left or right code)
- Double tick
- Feature will not end with a tick
- Ticks will not be applied at special points

Line Cleaning

- Cleaning angles (angle bisecting)
- Minimum resolution maintenance
- Line "back-up" edit
- Combinations of the above options
- Data culling based on line inclination factors

Symbol Specifications

- Specification file building and update
- Selection of specification file (multiple product files)
- Override to standard specifications (up to 10 overrides)

Input/Output Options

- LIS Table File (Input)
- GERBER 2032 Plotter (Output)
- Xynetics Plotter (Output)
- MMS-32/Raster Plotter Interface (Output)

Point Symbology

- Circle
- Dot
- Arrow
- Cross
- Half-Arrow
- Square
- Triangle
- Pyramid
- Arc/Chord (Mine Symbol)

Multiple Symbols

Various combinations of point and lineal symbology can be generated.

- Dash/Case
- Dash/Cross
- Dash/Dot
- Dash/Tick
- Dash/Circle
- Line/Arrow
- Line Center/Case

TABLE OF CONTENTS


| | | Page |
|---|---|---|
| I. | INTRODUCTION | I-1 |
| | A. Purpose | I-1 |
| | B. Organization | I-1 |
| II. | SYSTEM OPERATING ENVIRONMENT | II-1 |
| | A. System Characteristics | II-1 |
| | B. Processing Cycle | II-1 |
| | C. Software Modules | II-3 |
| | D. File Formats | II-6 |
| III. | PROGRAM DESCRIPTION | III-1 |
| | A. MONITR | III-1 |
| | B. SETUP | III-6 |
| | C. IPUT (LIS) | III-23 |
| | D. Subroutine JCRSE | III-32 |
| | E. Subroutine VECTAB | III-35 |
| | F. FORHED | III-45 |
| | G. Subroutine REDREC | III-51 |
| | H. OPUT (Gerber 2032 Plotter) | III-55 |
| | I. OPUT (XYNETICS Plotter) | III-61 |
| | J. OPUT - (MMS 32) | III-66 |
| | K. PAK298 | III-73 |
| | L. UNVHIS | III-80 |
| | M. OPUT (Dummy OPUT) | III-86 |
| | N. FETCOR | III-88 |
| | O. CKDESP | III-101 |
| | P. NOCORR | III-106 |
| | Q. SMOOTH | III-108 |
| | R. BACKUP | III-136 |
| | S. SIMBOL | III-144 |

TABLE OF CONTENTS
(continued)


| | | Page |
|---|---|---|
| T. | DASHER | III-156 |
| U. | Subroutine ABSPNT | III-165 |
| V. | FINDPT | III-168 |
| W. | POINTS | III-172 |
| X. | SPACE | III-183 |
| Y. | DOTTER | III-189 |
| Z. | CIRCLE | III-192 |
| AA. | Subroutine TICKER | III-199 |
| BB. | CASER | III-207 |
| CC. | BAKOVE | III-220 |
| DD. | CASEST | III-231 |
| EE. | CASEIT | III-235 |
| FF. | CPDIST | III-239 |
| GG. | CASPOT | III-242 |
| HH. | ARROW | III-246 |
| II. | SLOPE | III-254 |
| JJ. | CROSS | III-258 |
| KK. | SQUARE | III-264 |
| LL. | TRNGLE | III-270 |
| MM. | PYRMID | III-276 |
| NN. | ARCORD | III-281 |
| OO. | LINUP | III-289 |
| PP. | SPEC | III-291 |
| QQ. | HEADSUM | III-301 |

APPENDIX I - COBOL AND FORTRAN COMMON AREAS .... I-1
APPENDIX II - LIS FILE FORMAT ..... II-1
APPENDIX III - MMS-32 WORD FILE FORMAT FOR DMAAC. III-1

# LIST OF FIGURES


| Fig. No. | | Page |
|---|---|---|
| II-1 | GLSS Processing Cycle | II-2 |
| II-2 | Software Configuration | II-7 |
| III-1 | MONITR Process Flow | III-4 |
| III-2 | SETUP Process Flow | III-11 |
| III-3 | IPUT Process Flow | III-27 |
| III-4 | JCRSE Process Flow | III-34 |
| III-5 | VECTAB Process Flow | III-39 |
| III-6 | LIS Vector Code and Incremental Values | III-44 |
| III-7 | FORHED Process Flow | III-48 |
| III-8 | REDREC Process Flow | III-53 |
| III-9 | OPUT Process Flow | III-58 |
| III-10 | OPUT Process Flow | III-64 |
| III-11 | OPUT Process Flow | III-69 |
| III-12 | HIS Standard Format of Labeled Files | III-76 |
| III-13 | Two Hundred and Ninety Eight Words Per Physical Record Nine 32 - Word Records | III-77 |
| III-14 | PAK298 Process Flow | III-78 |
| III-15 | UNVHIS Process Flow | III-83 |
| III-16 | FETCOR Process Flow | III-96 |
| III-17 | CKDESP Process Flow | III-104 |
| III-18 | SMOOTH Process Flow | III-116 |
| III-19 | BACKUP Process Flow | III-140 |
| III-20 | SIMBOL Process Flow | III-149 |
| III-21 | DASHER Process Flow | III-160 |
| III-22 | ABSPNT Process Flow | III-167 |
| III-23 | FINDPT Process Flow | III-171 |
| III-24 | POINTS Process Flow | III-174 |
| III-25 | SPACE Process Flow | III-186 |
| III-26 | DOTTER Process Flow | III-191 |
| III-27 | CIRCLE Process Flow | III-197 |

LIST OF FIGURES
(continued)


| Fig. No. | | Page |
|---|---|---|
| III-28 | TICKER Process Flow..... | III-203 |
| III-29 | CASER Process Flow..... | III-212 |
| III-30 | BAKOVE Process Flow..... | III-225 |
| III-31 | CASEST Process Flow..... | III-233 |
| III-32 | CASEIT Process Flow..... | III-238 |
| III-33 | CPDIST Process Flow..... | III-241 |
| III-34 | CASPOT Process Flow..... | III-245 |
| III-35 | ARROW Process Flow..... | III-250 |
| III-36 | SLOPE Process Flow..... | III-256 |
| III-37 | CROSS Process Flow..... | III-262 |
| III-38 | SQUARE Process Flow ..... | III-268 |
| III-39 | TRNGLE Process Flow..... | III-274 |
| III-40 | PYRMID Process Flow ..... | III-279 |
| III-41 | ARCORD Process Flow..... | III-286 |
| III-42 | SPEC Process Flow..... | III-294 |
| III-43 | Feature Descriptor Codes ..... Stored Sequential Via Direct Access | III-298 |
| III-44 | Symbol Piece Specification Record Stored ... Randomly Via Direct Access File | III-299 |
| III-45 | Symbol Piece Type..... Equivalance | III-300 |

# LIST OF TABLES


| Table No. | | Page |
|---|---|---|
| III-1 | Symbol Specification Descriptor Code Format Sequential File File Code 08 | III-92 |
| III-2 | Symbol Specification Random Record..... Random File File Code 02 | III-93 |
| III-3 | Contents of IHEAD When Symbol Directives.. are Stored | III-94 |
| III-4 | Correlation between feature descriptors and . symbol specification directive random records | III-95 |
| III-5 | Symbol Piece Type and Associated Values ... | III-145 |

# SECTION I
# INTRODUCTION

## A. *Purpose*

The purpose of Volume II of the Final Technical Report is to describe the GLSS software in terms of the operating environment and definition of all software routines.

## B. Organization

Section II of this volume describes the software environment of GLSS, including general processing cycle, software configuration and summary, common data areas, and input and output file descriptions (see Appendix).

Section III presents individual program descriptions. Each program is described according to the following items:

- Functional Description
- Computer Definition
- Program Description
- Program Constants and Variables
- Error Conditions



PRECEDING PAGE BLANK-NOT FILMED

# SECTION II
# SYSTEM OPERATING ENVIRONMENT

## A. System Characteristics

### 1. Hardware

GLSS operates on the UNIVAC-1108 computer system and requires the following system resources and peripheral equipment: card reader or TTY for submission of job control/directives data; line printer for job reporting; two 9-track tape units for input and output of cartographic feature files and approximately 40K words of memory for job execution.

### 2. System Software

GLSS operates under the EXEC 8 operating environment and utilizes general system services of EXEC 8. Output of GERBER and Xynetics formatted files employs standard plotter subroutines which are resident on the UNIVAC 1108.

### 3. GLSS Software

GLSS consists of 43 programs and is programmed in ASCII COBOL and FORTRAN V. COBOL is employed for system control, job setup, and job reporting. FORTRAN V is used for all symbol application programs.

## B. Processing Cycle

The general processing cycle for a GLSS job execution is presented in Figure II-1.

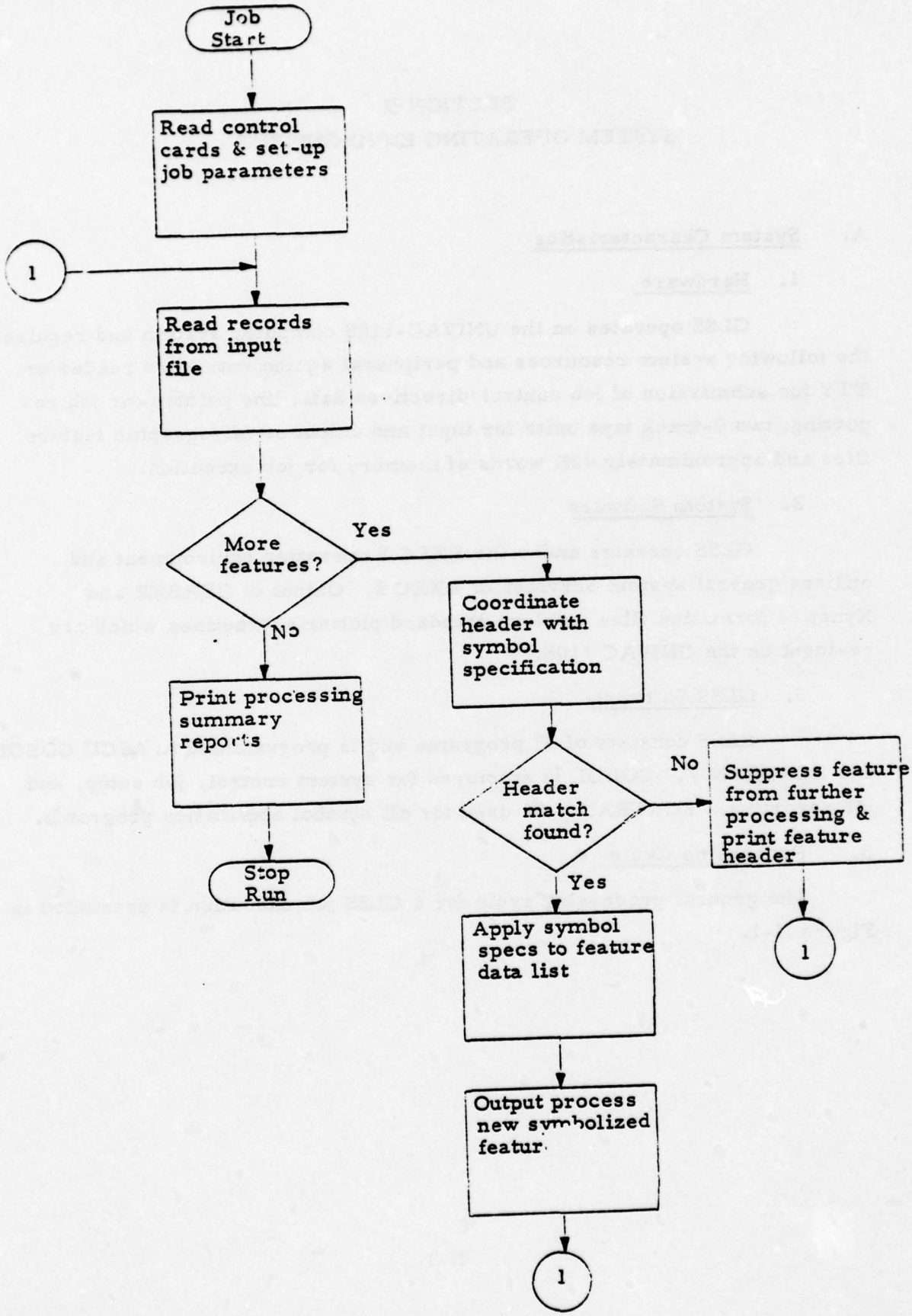



Figure II-1 GLSS Processing Cycle

## C. Software Modules

A summary of GLSS program, programming languages, and functional purposes are presented below. Configuration of the software is illustrated in Figure II-2.

| Program | Language | Purpose |
|---|---|---|
| ABSPNT | FORTRAN V | Checks current point against absolute point table. |
| ARCORD | FORTRAN V | Generates an arc and a chord for a more symbol. |
| ARROW | FORTRAN V | Computes two or three point which define an arrow or half-arrow. |
| BAKOVE | FORTRAN V | Determine if two line segments intercept & cleans the intersection. |
| BACKUP | FORTRAN V | Determines if the third of three adjacent points reverses direction & alters the data list accordingly. |
| CASER | FORTRAN V | Controls & performs the case processing. |
| CASEST | FORTRAN V | Computes two points perpendicular to a line segment and away from one of its end points. |
| CASEIT | FORTRAN V | Computes two points on a perpendicular bisect away from a line segment. |
| CASPOT | FORTRAN V | Computes two pairs of points which are parallel and equidistant from a line segment. |
| CKDESP | FORTRAN V | Compares file feature header against specification headers. |
| CIRCLE | FORTRAN V | Generates a circle of a given size about a center point. |

| Program | Language | Purpose |
|---|---|---|
| CPDIST | FORTRAN V | Computes & compares the distance between two points with minimum distance. |
| CROSS | FORTRAN V | Generates four points defining a cross about a center point. |
| DASHER | FORTRAN V | Generates dashed line symbology. |
| DOTTER | FORTRAN V | Generates a dot at a point. |
| FETCOR | FORTRAN V | Retrieves specification files and examine headers for a match. |
| FINDPT | FORTRAN V | Compute a point between two points at a given distance from the first point. |
| FORHED | FORTRAN V | Formats LIS headers for GLSS processing. |
| HEADSUM | FORTRAN V | Reads an LIS Table File and prints a summary of headers and tally of data points. |
| IPUT (LIS INPUT) | FORTRAN V | Reads & formats LIS Table file. |
| JCRSE | FORTRAN V | Formats a JCRS or JCRE for MMS-32 file formating. |
| LINUP | ASCII COBOL | Formats & prints the processing summary reports. |
| MONITR | ASCII COBOL | Control the sequencing of processes for the symbolization execution. |
| NOCORR | FORTRAN V | Prints headers of those features which no specifications could be located. |
| OPUT | FORTRAN V | Output processes feature data to a MMS-32 formated file. |

| Program | Language | Purpose |
|---|---|---|
| OPUT (DUMMY) | FORTRAN V | Prints a formated dump of all data points passed to the output phase. |
| OPUT (GERBER) | FORTRAN V | Output processes symbolized feature data to a GERBER 2032 formated file. |
| OPUT (XYNETICS) | FORTRAN V | Output processes symbolized feature data to a Xynetics formated file. |
| PAK298 | FORTRAN V | Packs MMS-32 word records into a 298 word buffer. |
| POINTS | FORTRAN V | Locates two points along the data list which are a specified distance from the current point. |
| PYRMID | FORTRAN V | Computes three points defining a pyramid. |
| REDREC | FORTRAN V | Read a LIS table coordinate file. |
| SETUP | ASCII COBOL | Reads input data cards & sets up job processing directives. |
| SIMBOL | FORTRAN V | Controls all symbol application processing. |
| SLOPE | FORTRAN V | Computes the approximate slope of the line at a given point. |
| SMOOTH | FORTRAN V | Performs various level of data reduction, line clearing, and smoothing processes. |
| SPACE | FORTRAN V | Walks down a data list a specified distance. |
| SPEC | FORTRAN V | Read symbol specification data cards & builds/updates disc resident specification files. |
| SQUARE | FORTRAN V | Computes four points defining a square centered on a center point. |

| Program | Language | Purpose |
|---|---|---|
| TICKER | FORTRAN V | Generates full and half ticks along a data list. |
| TRNBLE | FORTRAN V | Computes three points which defines an equilateral triangle. |
| UNVHIS | FORTRAN V | Converts floating point numbers from UNIVAC 1108 to UIS 6000/600 formats. |
| VECTAB | FORTRAN V | Converts LIS vector data to absolute coordinate values. |

D. File Formats

The input file to GLSS is a standard LIS Table File presented in Appendix I. Internal processing of feature data and symbols is performed via eight common data areas whose COBOL and FORTRAN formats are defined in Appendix II. References to common areas in the following program documentation frequently refer to common areas -- C1, C2, ...

| | |
|---|---|
| C1 | Feature Descriptor Data |
| C2 | Feature Line Center Data |
| C3 | Symbol Spec Directive |
| C4 | Symbol Spec Directive Override |
| C5 | Status Indicator, Flags, and Pointers |
| C6 | Parameters and Variables |
| C7 | Process Tally Summary Report |
| C8 | LIS |

Output files generated by GLSS include GERGER, Xynetics, and MMS-32. GERBER and Xynetics Plotter files are standard formats produced by calls to DMAAC subroutines. The MMS-32 format is defined in Appendix III.

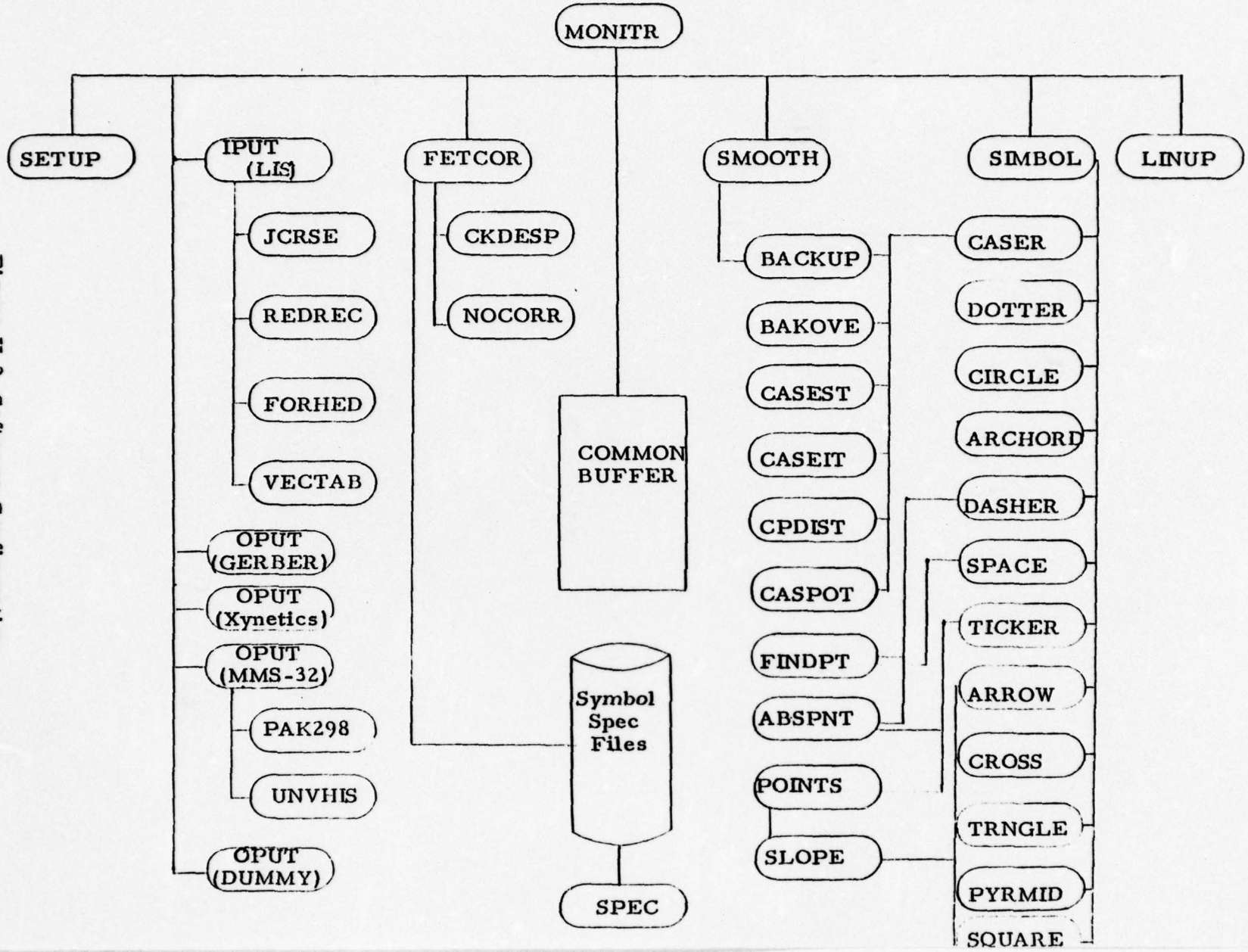



Figure II-2 Software Configuration

# III. PROGRAM DESCRIPTION

## A. MONITR

### 1. Functional Description

MONITR controls the sequence of operations performed during a symbolization job run. Job operations controlled by MONITR include job set-up, file input and output processing, correlation of symbol specifications to features, data smoothing, symbol application, and job summarizing and wrap-up.

### 2. Computer Definition

#### a. Core Memory Used

164 octal words.

#### b. Peripheral Equipment

None

### 3. Program Description

#### a. Calling Routines

None

#### b. Subroutines Used

SETUP
IPUT
FETCOR
SMOOTH
SIMBOL
OPUT
LINUP

#### c. Input

None



PRECEDING PAGE BLANK-NOT FILMED

d. Output

None

e. Processing Methodology

MONITR initiates the job by calling SETUP for reading of control cards and setting of job parameter flags and indicators. Input is then called to process records from the input file. If a file start, registration, or file end record was input, the OPUT Module is called for outputting the record. If a new feature is input, FETCOR is then called to determine the type of symbology to be applied. If the user requested the file to be smoothed or if the feature is to be symbolized (other than line center), the SMOOTH Routine is called. The Symbol Controller (SIMBOL) is then called for symbolization processing. On return to MONITR, a symbol segment will be output processed, point tallies updated for line center features, and control returned to SIMBOL (CALLBACK Flag set) or a new feature segment will be input processed. A processing summary report will be generated on the line printer when the job is completed. Refer to Figure III-1 for a process flow diagram of MONITR.

f. Calling Sequence

Not applicable.

g. Major Algorithms

None

4. Program Constants and Variables

Program variables interrogated by MONITR include:

NUM-SUM-PIECES (=1 and SYM-TYPE = 1 call OPUT)
SYM-TYPE ( =1 and NUM-SYM-PIECES = 1 call OPUT)
STORE-SPEC-HEAD ( =1 call OPUT)

SYM-READY-OUT ( ≠ 0 call OPUT)
SYMBOL-CALLBACK ( ≠ 0 call SYMBOL)
ABORT-JOB-ID ( ≠ 0 call LINUP and stop run)
JOB-THRU-FLAG ( ≠ 0 call LINUP and stop run)
SMOOTH-TYPE ( ≠ 0 call SMOOTH)
NEW-FEAT ( ≠ 0 call FETCOR)

5. Error Conditions

ABORT-JOB-ID ≠ 0 the run is aborted.

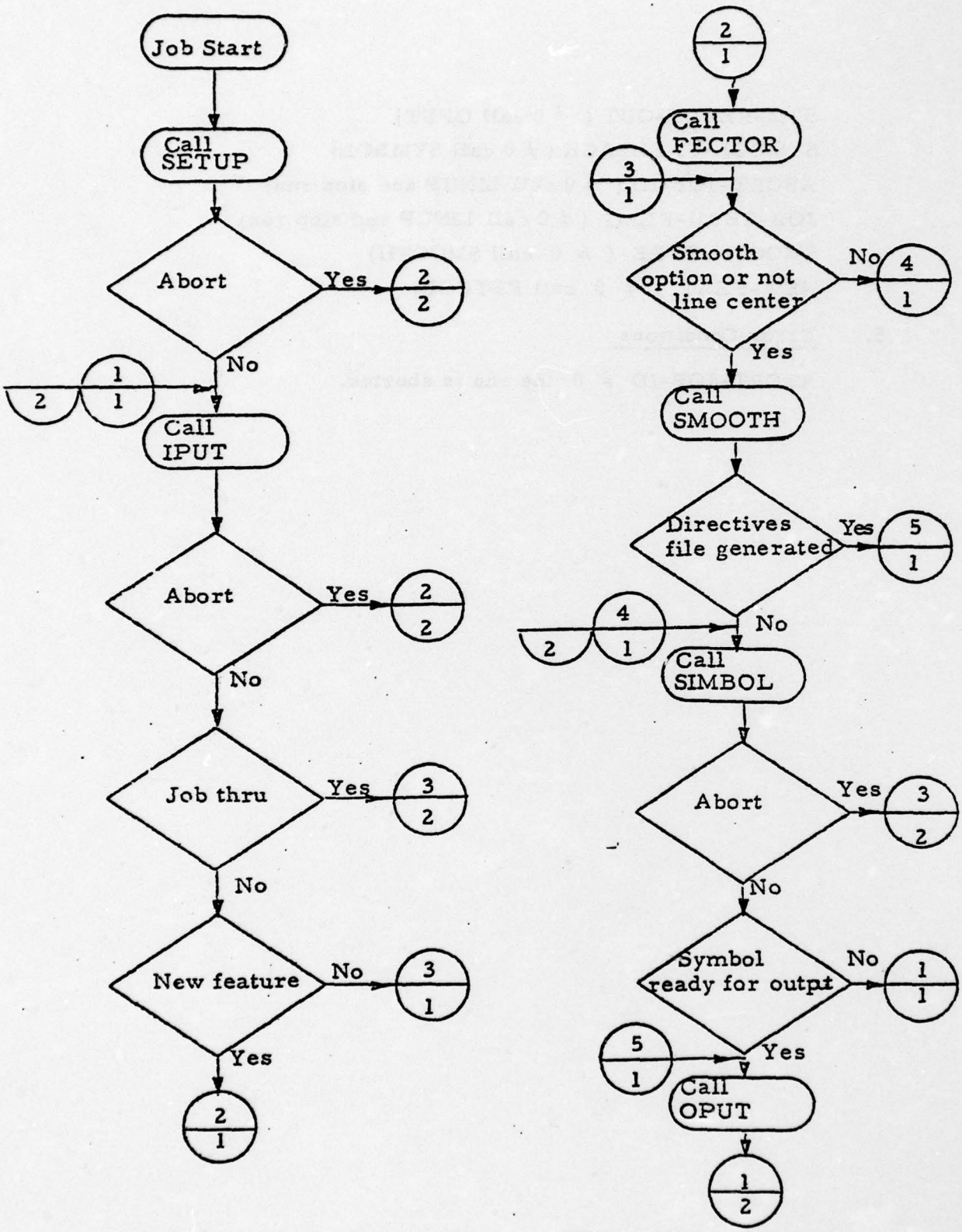



Figure III-1 MONITR Process Flow (Page 1 of 2)

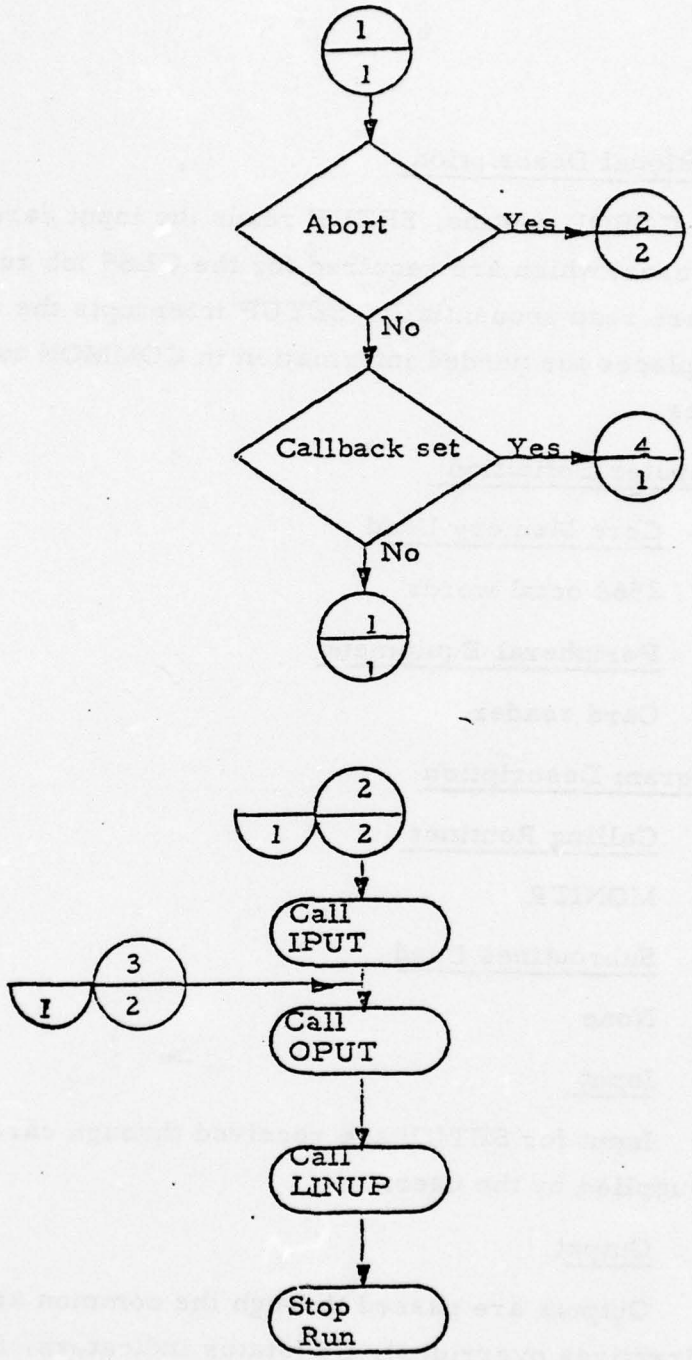



Figure III-1 MONITR Process Flow (Page 2 of 2)

B. SETUP

1. Functional Description

As a COBOL routine, SETUP reads the input cards (supplied by the user) which are required for the GLSS job run. As the input cards are read sequentially, SETUP interrupts the input, as necessary, and places the needed information in COMMON to be used by the other routines.

2. Computer Definition

a. Core Memory Used

2562 octal words

b. Peripheral Equipment

Card reader.

3. Program Description

a. Calling Routines

MONITR

b. Subroutines Used

None

c. Input

Input for SETUP are received through cards in the job stream as supplied by the user.

d. Output

Outputs are passed through the common area C4 (symbol spec directives overrides), C5 (status indicators, flags and pointers), and C6 (parameters and variables).

e. Processing Methodology

As a COBOL routine, SETUP, upon call from MONITR, will input the GLSS job control cards supplied by the user. SETUP will proceed in the following way by reading input cards sequentially and checking card format. First, card one will be read, interpreted, and placed in common area C6 as the input file format (IN-FILE). Secondly, card two will be read, interpreted, and also placed in common area C6 as the output format file (OUT-FILE). Thirdly, card three is then read, and determination is made as to whether or not the GLSS job is a special run to write a special MMS file which contains the symbol directives in the header recorder. The flag STORE-SPEC-HEAD is correspondingly set ( 0 or 1). If SETUP determines that the GLSS job is to be a special run, as described above, SETUP will look for input card nine, as described below, as the next input card. The fourth input card will be read next to determine which smooth option will be used in the GLSS run. The option which is determined is stored in SMOOTH-TYPE. Proceeding with a standard GLSS job, the fifth, sixth, and seventh input cards (minimum distance, maximum distance, and slope distance values) will sequentially be read, interpreted, and correspondingly be placed into common (MIN-DIST, MAX-DIST, and SLOPE-DIST). Next, input card eight is read to determine whether or not the input file contains the symbol directives in the header (YES or NO). The flag READ-SPEC-HEAD is set to show the status of the header (0 if NO, and 1 if YES). Following the eighth input card, or third input card (as described above), the nineth input card is read. This input card informs SETUP of the existence of following override input cards (YES or NO). If NO, SETUP returns control to the calling routine; if YES, SETUP proceeds to input the override cards and places the information obtained from the input cards into common area C4 (SYMBOL-SPEC-DIR-OVERRIDE). After reading all the override input cards, SETUP returns control to the calling routine. Refer to the User's Manual, and Symbol Specification

Build/Update subroutine SPEC for more detailed information. See Figure III-2 for processing flow diagram of SETUP.

f. Calling Sequence

Call SETUP

g. Major Algorithms

None

4. Program Constants and Variables

| | | |
|---|---|---|
| ABORT-JOB-ID | - | flag to abort the GLSS job run<br>=0 : do not abort<br>=1 : abort |
| CARD-NO | - | user's input control card sequence number |
| CARD-NUMBER | - | index of user's input control card |
| CARD-TOTAL | - | user's input control card count |
| END-CARD | - | used to test the user's end input control cards |
| IN-FILE | - | description of the input data format |
| INPUT-CARD | - | field in which the user's control input cards are read |
| MAX-DIST | - | contains the value used as the maximum resolution for trace data |
| MIN-DIST | - | contains the value used as the minimum resolution |
| NUM-OVER | - | counter of the total number of symbol overrides |
| OUT-FILE | - | description of the output data format |
| READ-SPEC-HEAD | - | status flag for reading the symbol specification from the data header record |
| SLOPE-DIST | - | contains the value which will be used to approximate the slope |
| SMOOTH-TYPE | - | integer value for describing the smooth option selected |

| | | |
|---|---|---|
| STNO1-OVER | - | array containing the sheet number one code for symbol spec overrides |
| STNO2-OVER | - | array containing the sheet number two code for symbol spec overrides |
| STORE-SPEC-HEAD | - | status flag for storing the symbol specification in the data header recorder |
| SYMBOL-CODE | - | integer numerical symbol piece code |
| SYM-CON-NON | - | array containing symbol piece conformal (0) or nonconformal (1) indicator for symbol piece spec overrides |
| SYM-FC1 | - | array containing feature class, type, and subtype for each symbol overrides |
| SYM-FC2 | - | array containing feature's first six codified descriptors for each of the symbol overrides |
| SYM-FC3 | - | array containing feature's last two codified descriptors for each of the symbol overrides |
| SYM-LW-OVER | - | array containing the symbol piece line weight for each symbol piece override |
| SYM-SZ-OVER | - | array containing the symbol piece size for each symbol piece override |
| SYM-TYPE-OVER | - | array containing the numerical code for symbol type for each symbol piece override |
| VAL | - | used to determine input distance values for minimum, maximum, and slope |
| VERDEX | - | index for symbol spec overrides |

5. Error Conditions

- o No data was found or ID was not in column 1 (ERR-1).
- o Input card missing or input card out of sequence (ERR-2).

- Input data card 3, 8, or 9 is incorrect (ERR-3).
- Premature END card found in override input cards (OVERRIDE-ERR-1).
- Premature EOSYM found in override input cards (OVERRIDE-ERR-2).
- Number of override symbol directives exceeds limit (OVERRIDE-ERR-3).
- Symbol data for override directive exceeds limit (OVERRIDE-ERR-4).
- Symbol override data is in incorrect format (OVERRIDE-ERR-5).

Errors occurring in the above situations will set the abort flag.

- No END data card, GLSS generated the end card. This error will not set the abort flag.

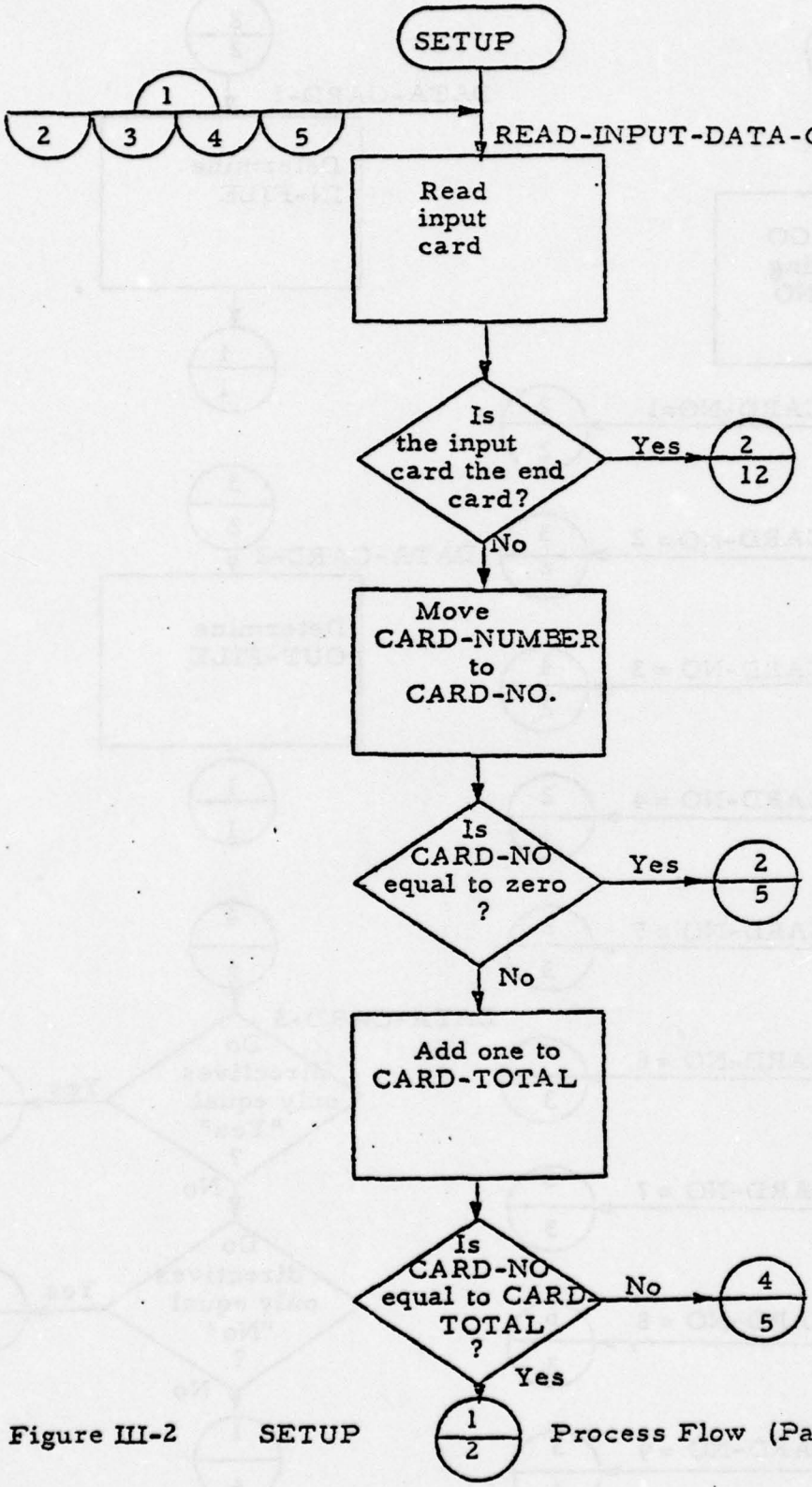



Figure III-2 SETUP Process Flow (Page 1 of 12)

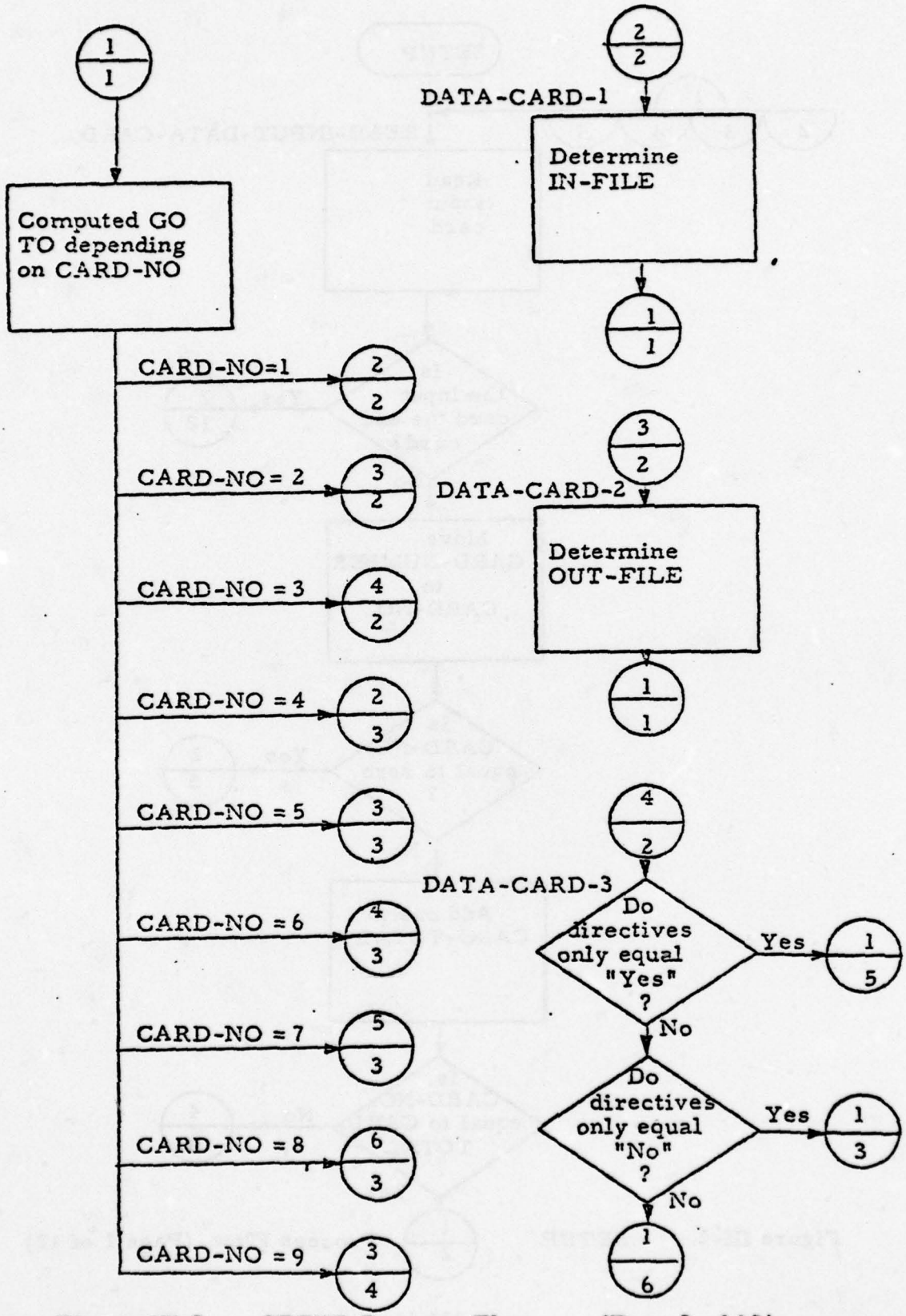



Figure III-2 SETUP Process Flow (Page 2 of 12)

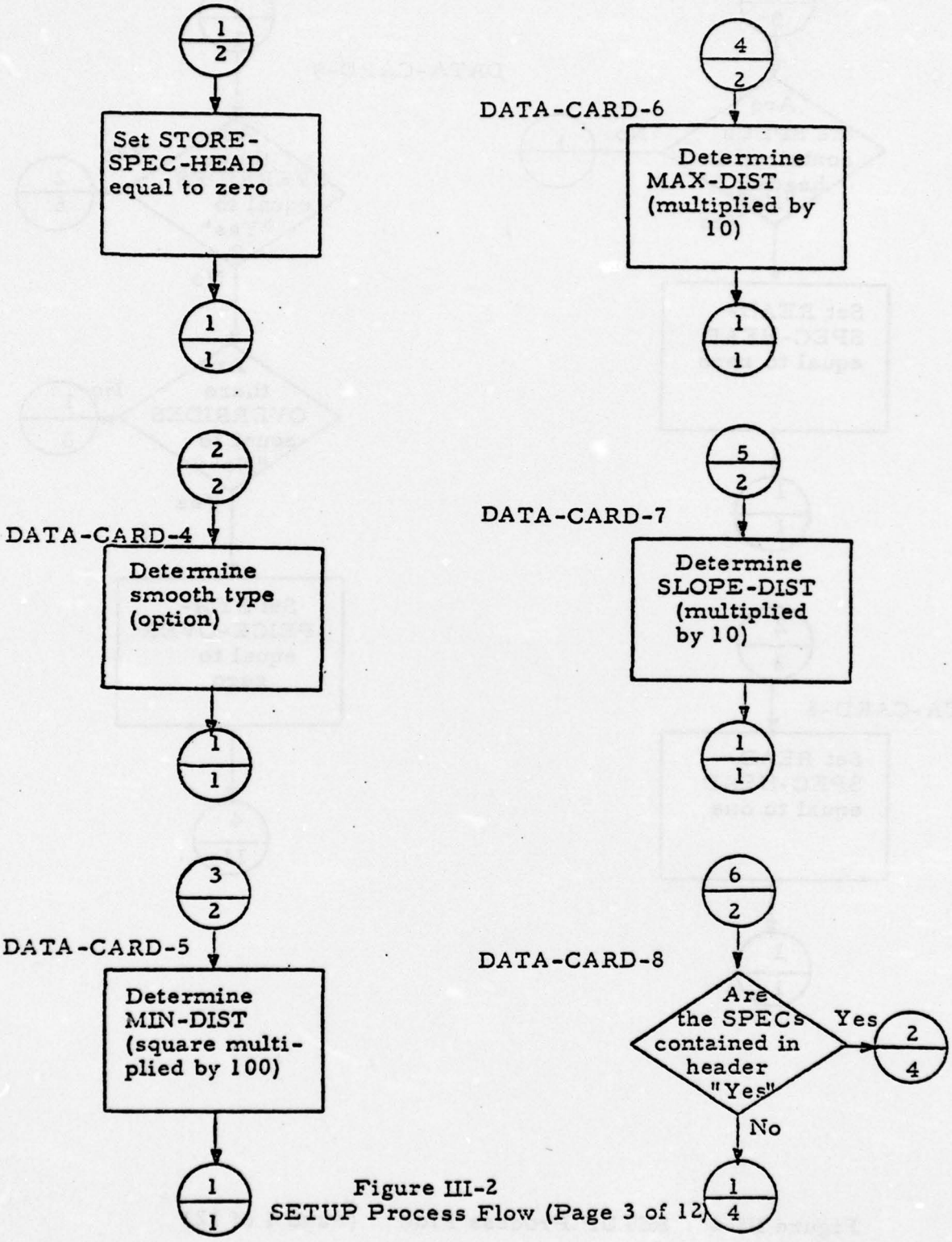



Figure III-2
SETUP Process Flow (Page 3 of 12)

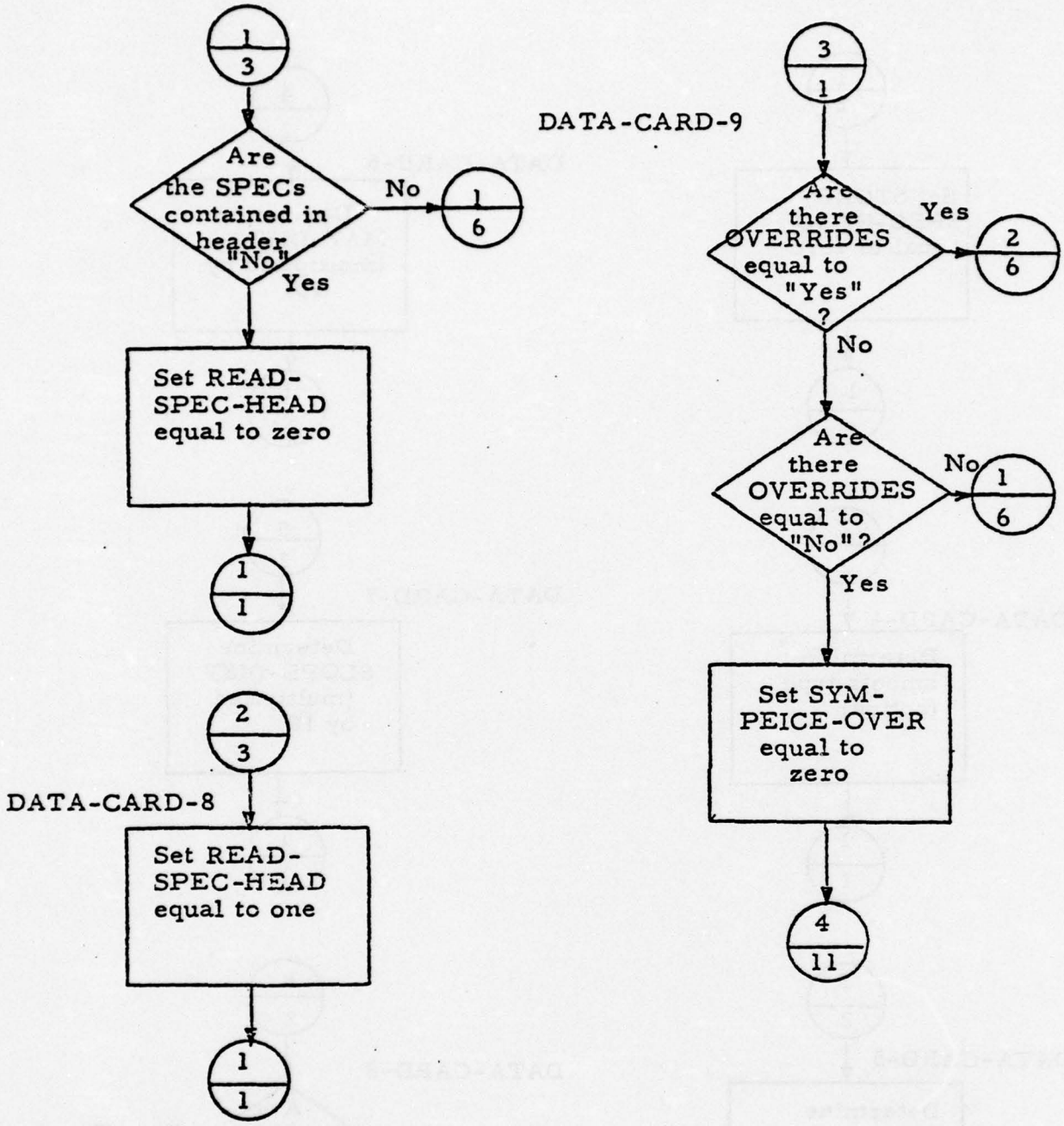



Figure III-2 SETUP Process Flow (Page 4 of 12)

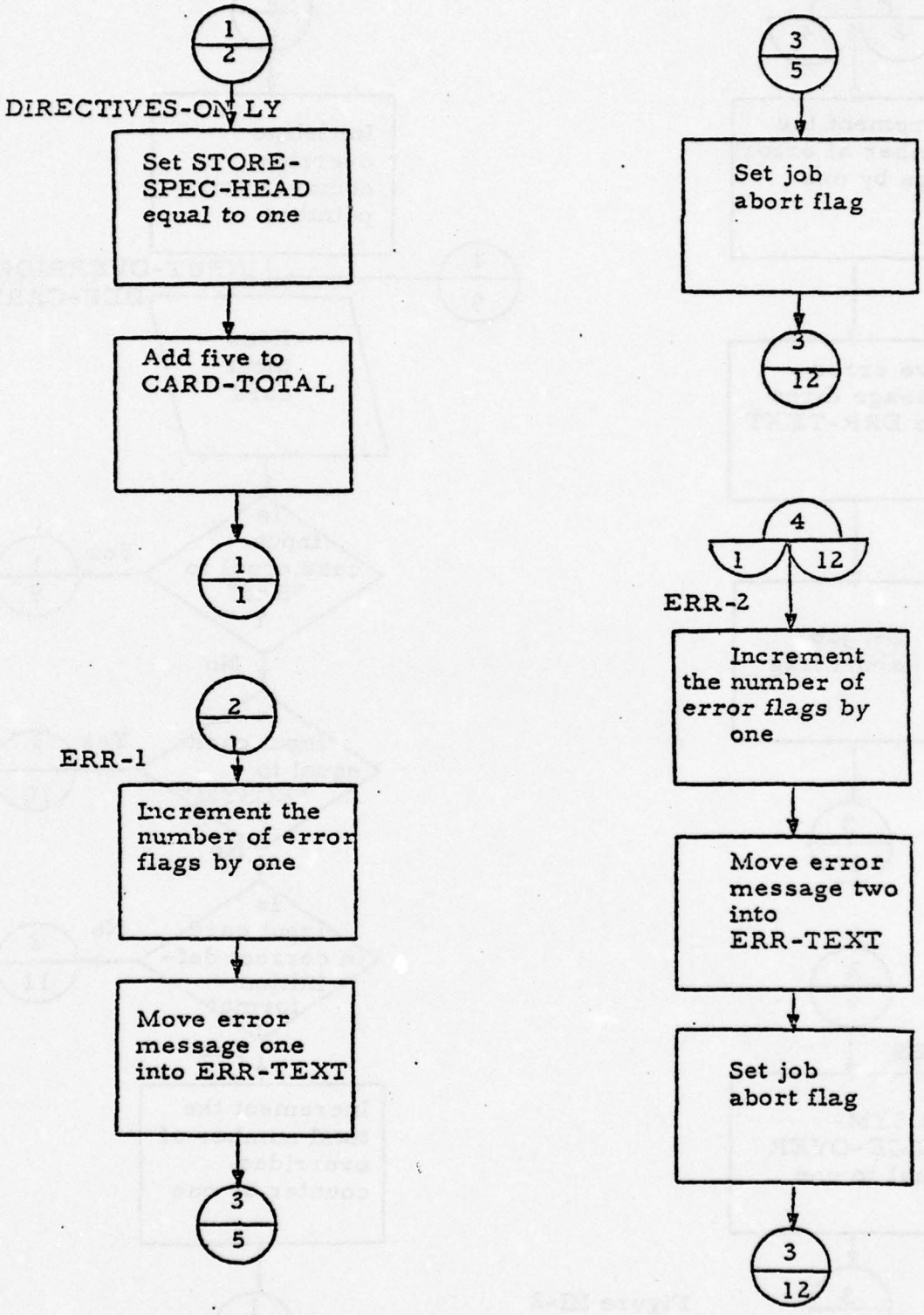



Figure III-2 SETUP Process Flow (Page 5 of 12)

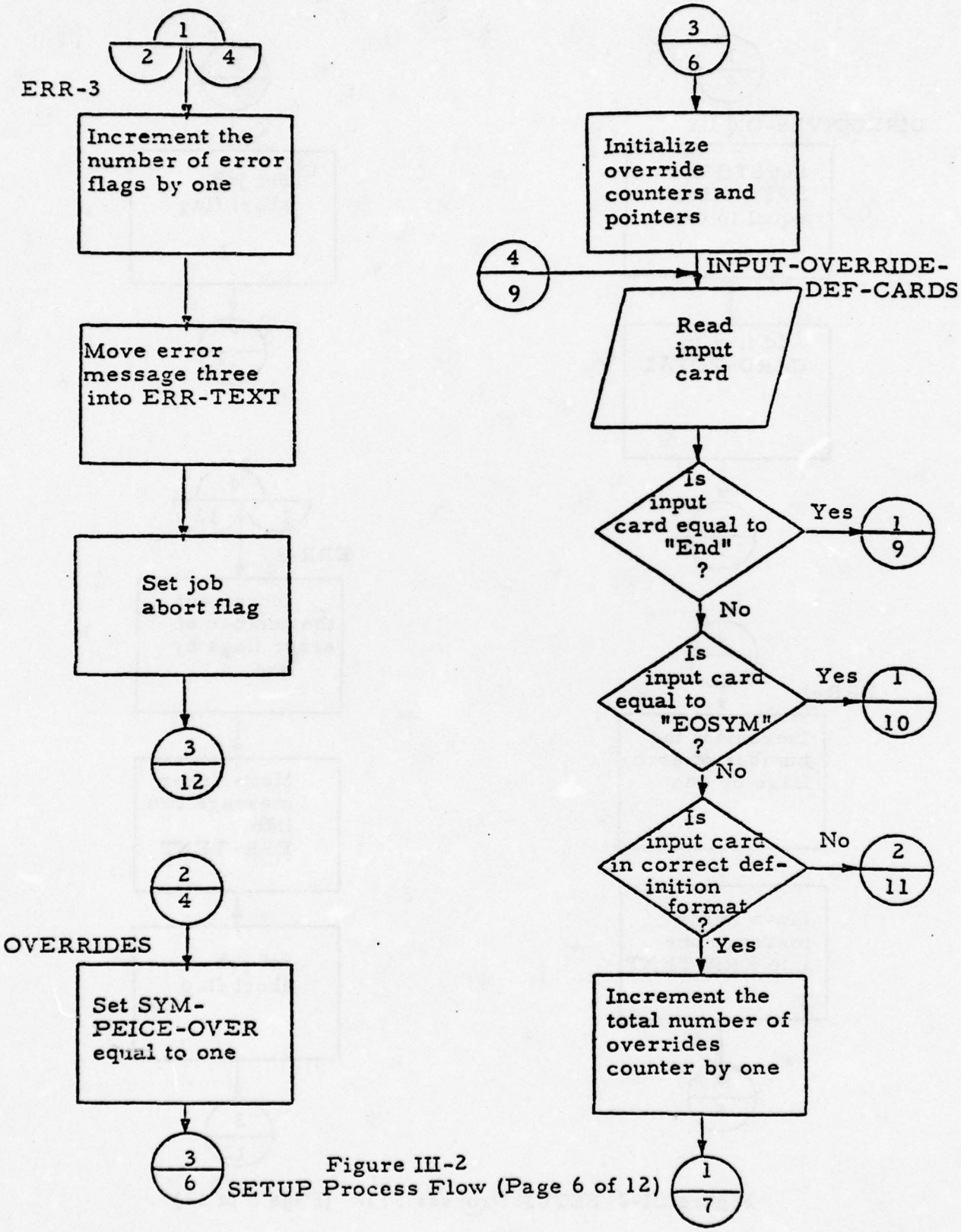



Figure III-2
SETUP Process Flow (Page 6 of 12)

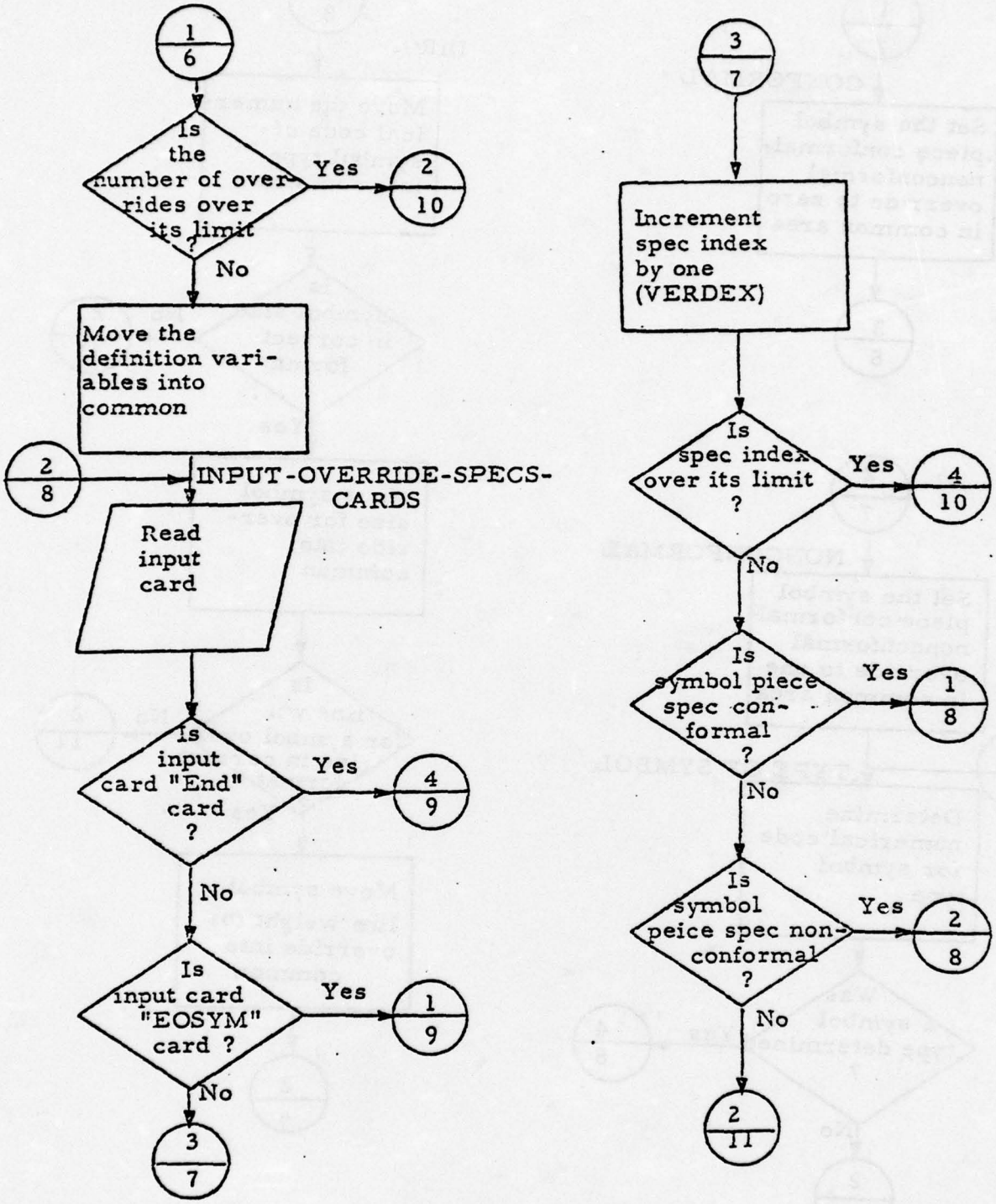



Figure III-2 SETUP Process Flow (Page 7 of 12)

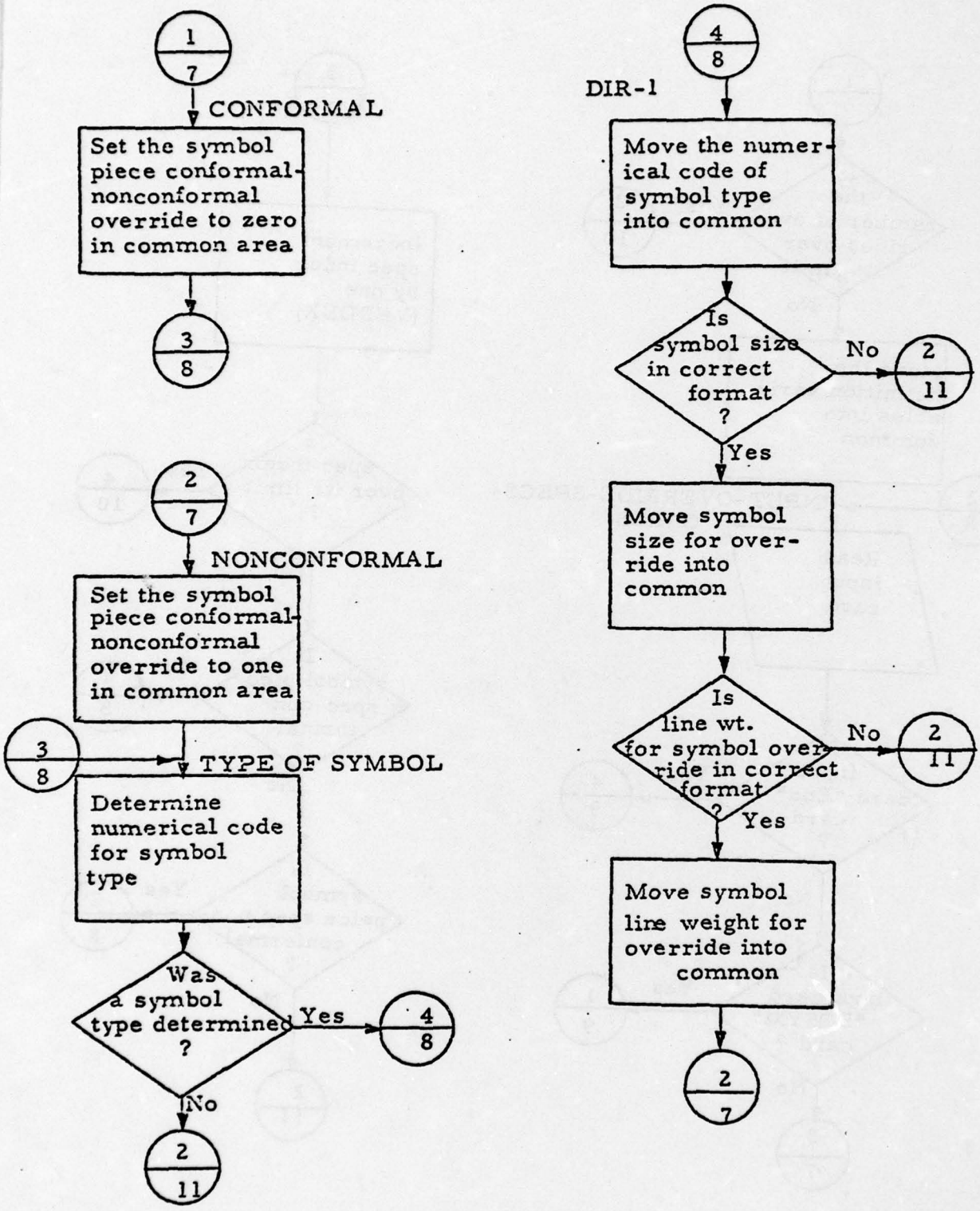



Figure III-2 SETUP Process Flow (Page 8 of 12)

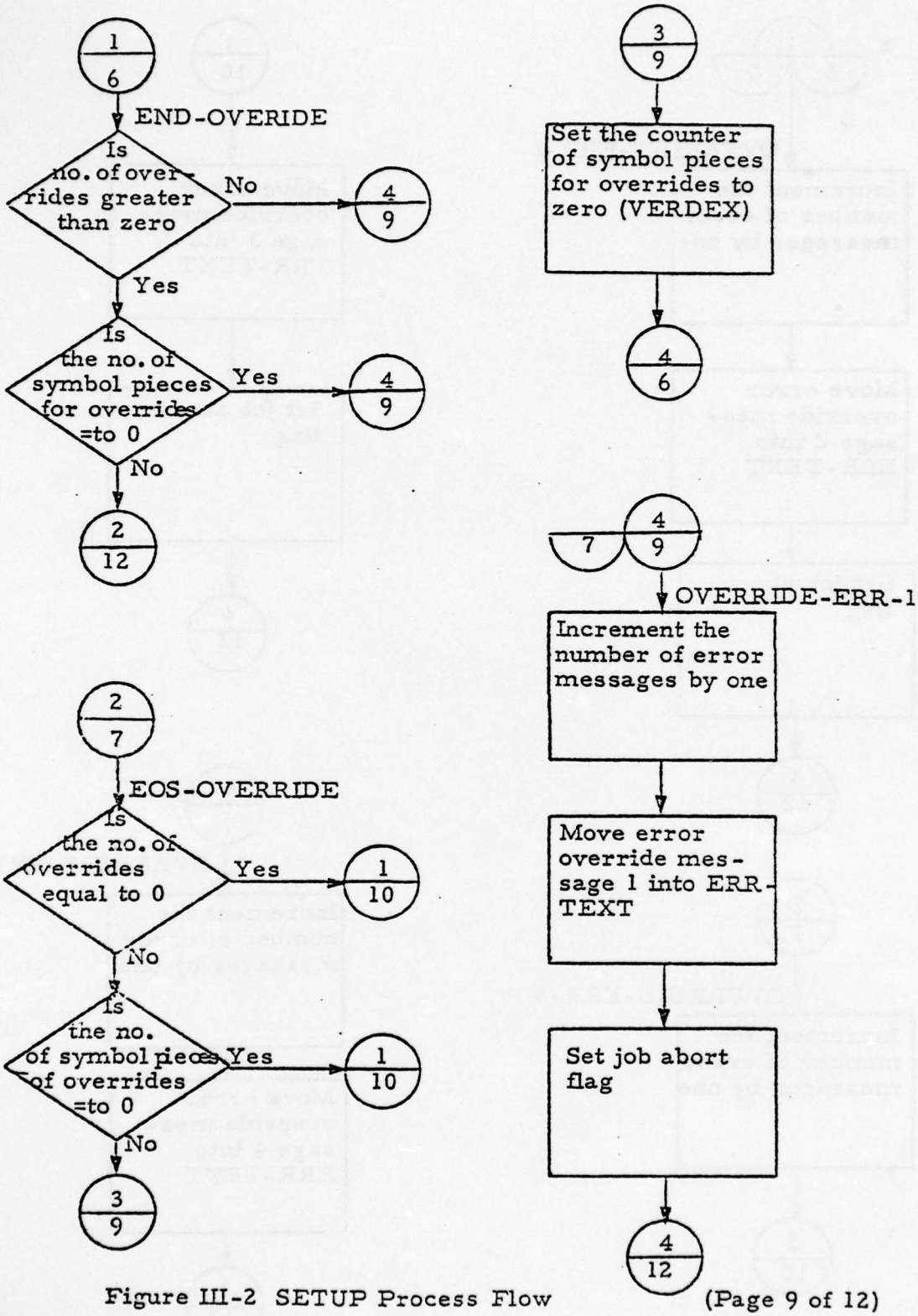



Figure III-2 SETUP Process Flow (Page 9 of 12)

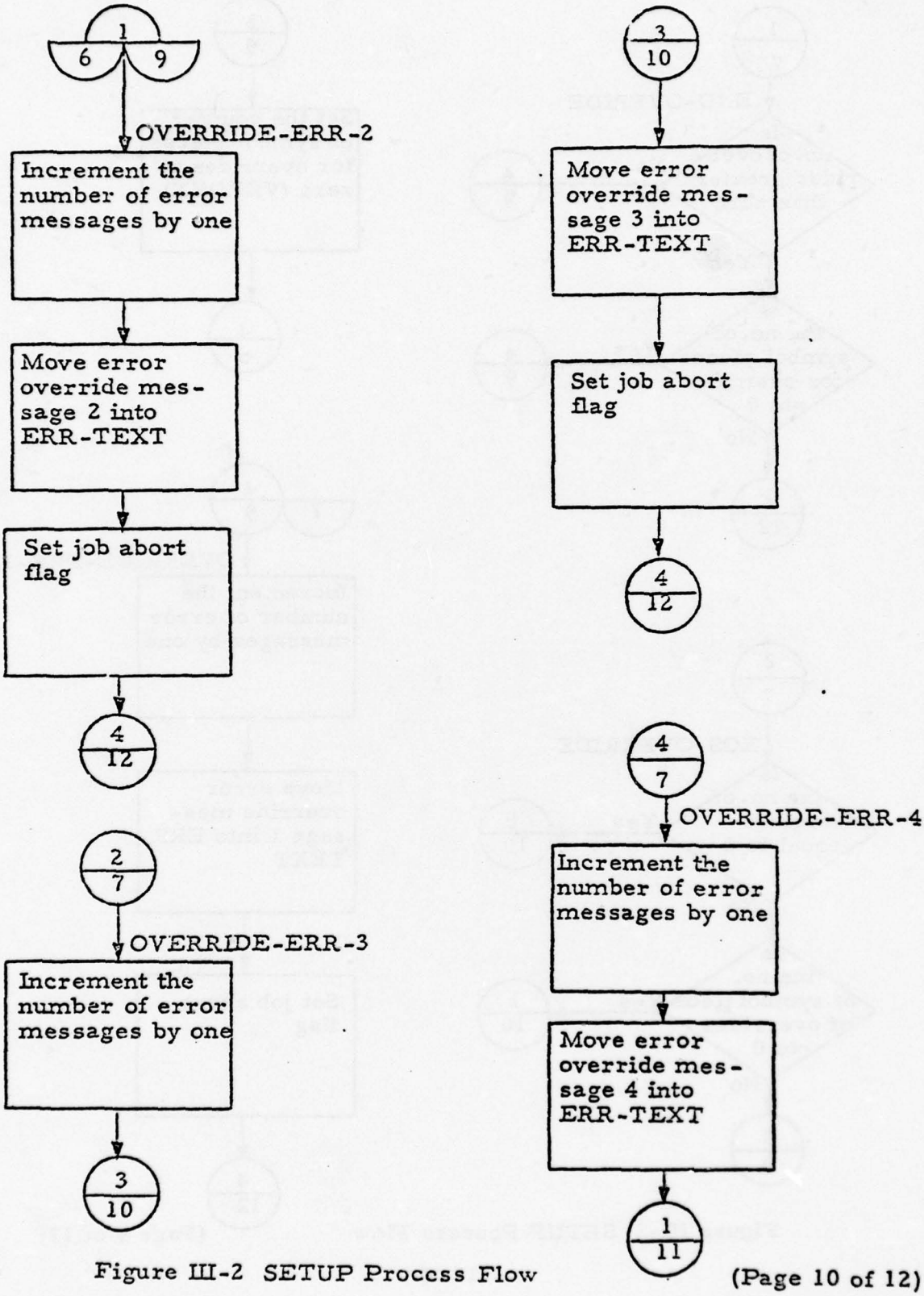



Figure III-2 SETUP Process Flow (Page 10 of 12)

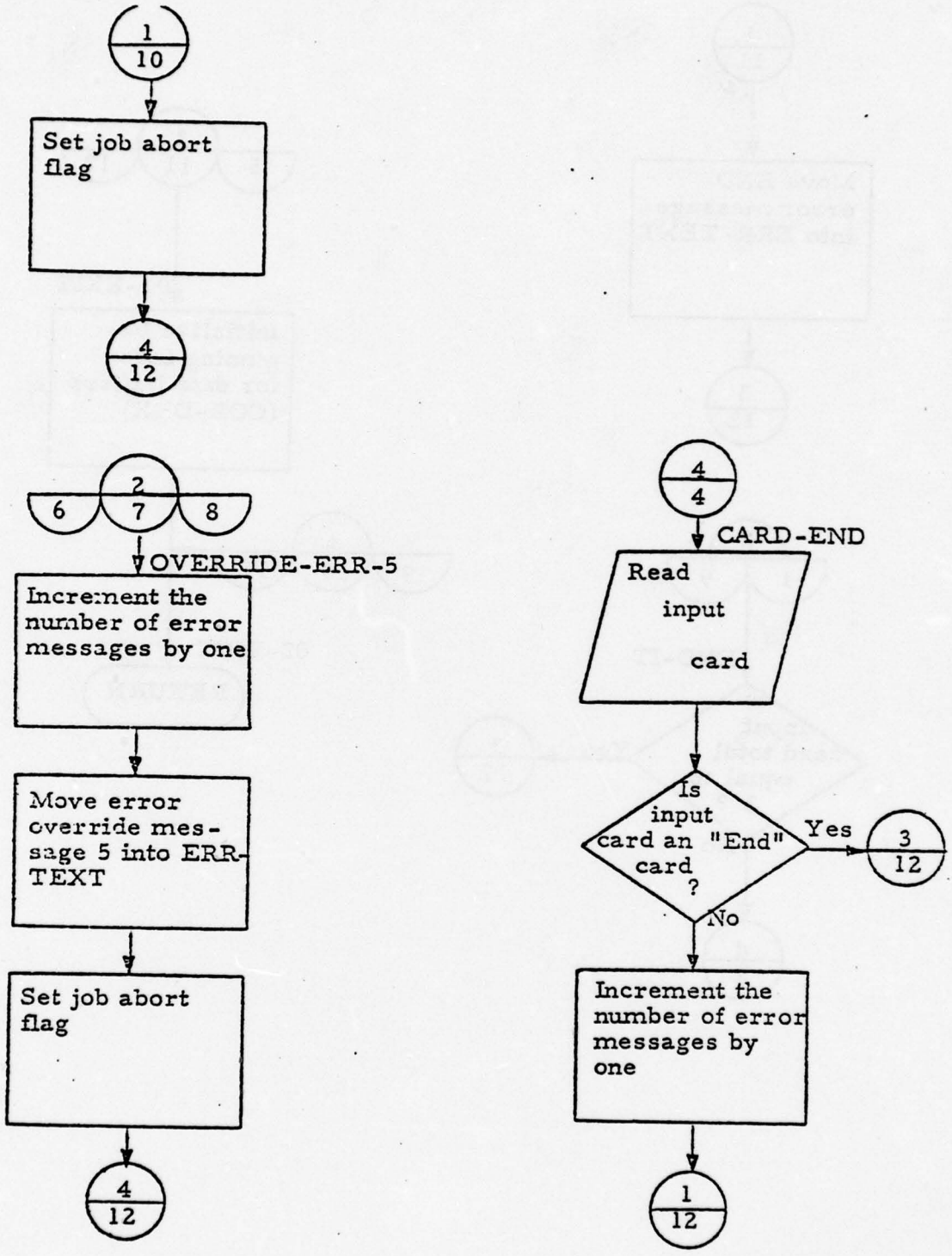



Figure III-2 SETUP Process Flow (Page 11 of 12)

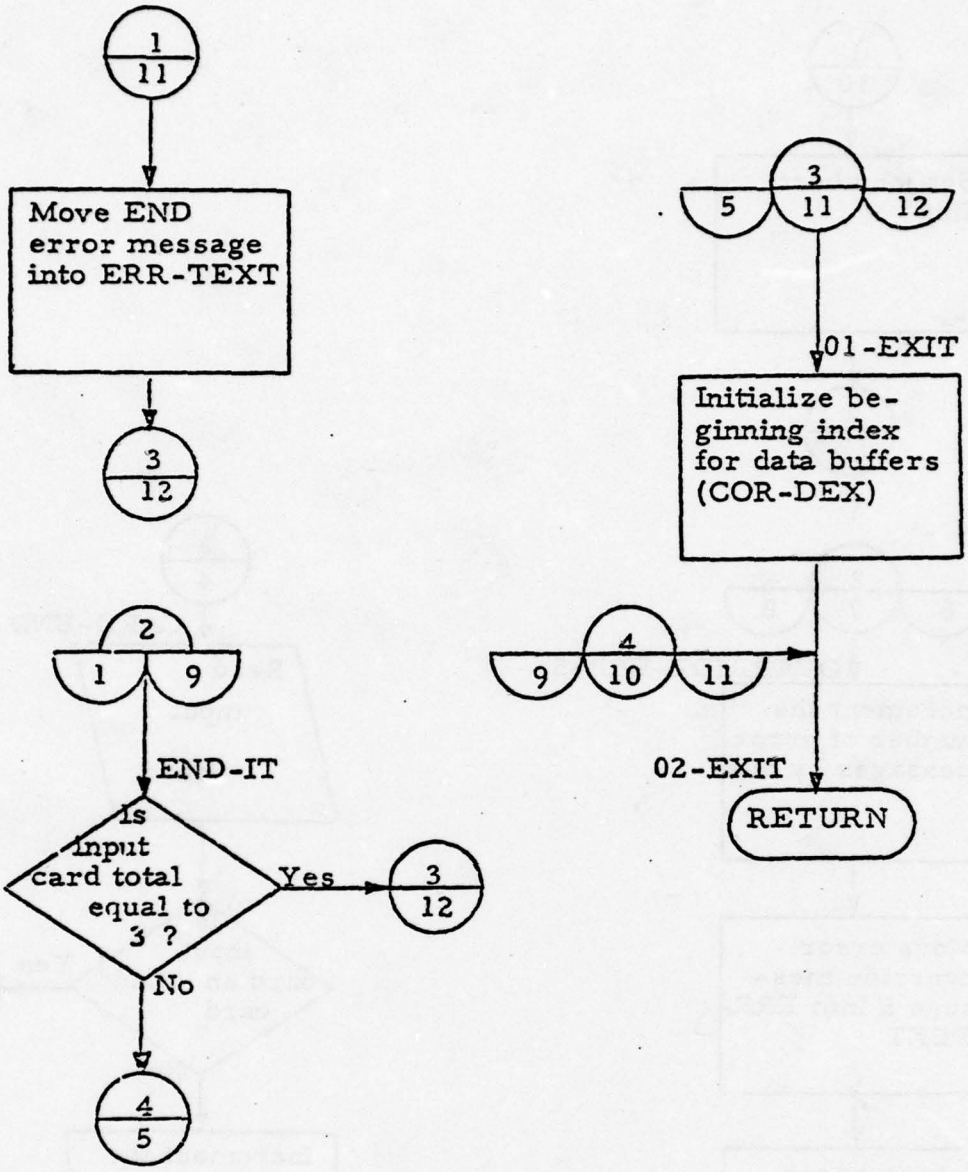



Figure III-2 SETUP Process Flow (Page 12 of 12)

C. IPUT (LIS)

1. Functional Description

The primary function of subroutine IPUT is to control the reading and converting of a Lineal Input System (LIS) table coordinate magnetic tape. The reading is accomplished via subroutine REDREC while the converting of the header data and table vector data is accomplished via subroutine FORHED and VECTAB, respectively. A secondary function of IPUT is to locate and report LIS record sequence errors.

2. Computer Definition

a. Core Memory Used

457 octal words

b. Peripheral Equipment

Not applicable.

3. Program Description

a. Calling Routine

MONITR

b. Subroutines Used

JCRSE
REDREC
FORHED
VECTAB

c. Input

Input consists of common area LIS, namely buffer IREC (buffer containing LIS record), mnemonics IRTYPE (record type), IBLNUM (block number), and NUMVEC (number of vectors in block). Other input is found in common area C5 (status indicators flags

and pointers) mnemonics IFTCNT (feature continuation flag).

d. Output

Output from subroutine IPUT is found in common area C5 mnemonic IJBEND.

e. Processing Methodology

Processing flow of subroutine IPUT is depicted in Figure III-3. Entry is made via GLSS control routine MONITR. A computed go to statement is executed with a resultant of one of the five options being taken. If first entry, the output format is examined (mnemonic IOFILE). If found to be the MMS-32 word type a start record (JCRS) is formatted in buffer IXYZ via subroutine JCRSE and process control returned to the calling routine MONITR. If the format is not the MMS-32 word type or upon the second entry, subroutine REDREC is called. If the record type read (mnemonic IRTYPE) is not zero or a premature end of file is reached, an error diagnostic is reported with the job being aborted. If the record type is zero the type of coordinate indicator is extracted from the input record and stored in mnemonic ITABLE. If ITABLE is not one, an error diagnostic indicating such is reported and again the job is aborted. If the coordinate indicator is one (table data), the next record is read from the input tape via REDREC. If this record is not a record type twenty, an error diagnostic is reported and the job aborted. If it is a record type twenty the recording resolution (microns), at which the trace data in the file was digitized, is extracted from buffer IREC and stored (IRSUIT). The next two record type twentys are stripped from the tape and the first header record (record type thirty) is read via REDREC. Subroutine FORHED is called to extract and reformat the input header data to the GLSS internal buffer common area C1. The next record (record type thirty one) is read via REDREC. This record type is the data list record for table coordinate files and contains incremental vectors. Subroutine VECTAB is then

called to change the above vectors to absolute table coordinate data and store this data into the GLSS coordinate data buffer (IXYZ). Record type thirty ones are processed until the last data block is reached for that feature or the feature continuation flag is set by VECTAB. When this occurs, an appropriate re-entry option is set with control being returned to the calling routine MONITR. Upon receiving process control re-entry option three or four is executed. If re-entry option three (mnemonic M1=3) is set the above header data record (record type thirty) read is executed along with the associated process. If re-entry option four (M1= 4) is set, control is returned to subroutine VECTAB to continue processing the previous data block. The above processes are repeated until a record type ninety (LIS file summary record) is reached indicating end of data. If the output format (mnemonic IOFILE) is the MMS-32 word an end record (JCRE) is formatted in buffer IXYZ via subroutine JCRSE with process control being returned to MONITR.

f. Calling Sequence

Call IPUT

g. Major Algorithm

None

4. Program Constants and Variables

IOFILE - type of format of output file BCD
IXYZ - buffer containing X, Y, coordinates
IRTYPE - input record type
ITABLE - type of coordinates on file (1 = table coordinates)
IRSUIT - recording resolution of digitized (microns)

5. Error Conditions

a. "PREMATURE END OF FILE ON L.I.S. INPUT TAPE".
b. "NO L.I.S. RECORD TYPE ZERO FOUND ON INPUT TAPE".

c. "NOT TABLE COORDINATE (VECTOR) L.I.S. INPUT TAPE".

d. "NO L.I.S. RECORD TYPE 20 FOUND ON INPUT TAPE".

e. "NO L.I.S. RECORD TYPE 90 FOUND ON INPUT TAPE".

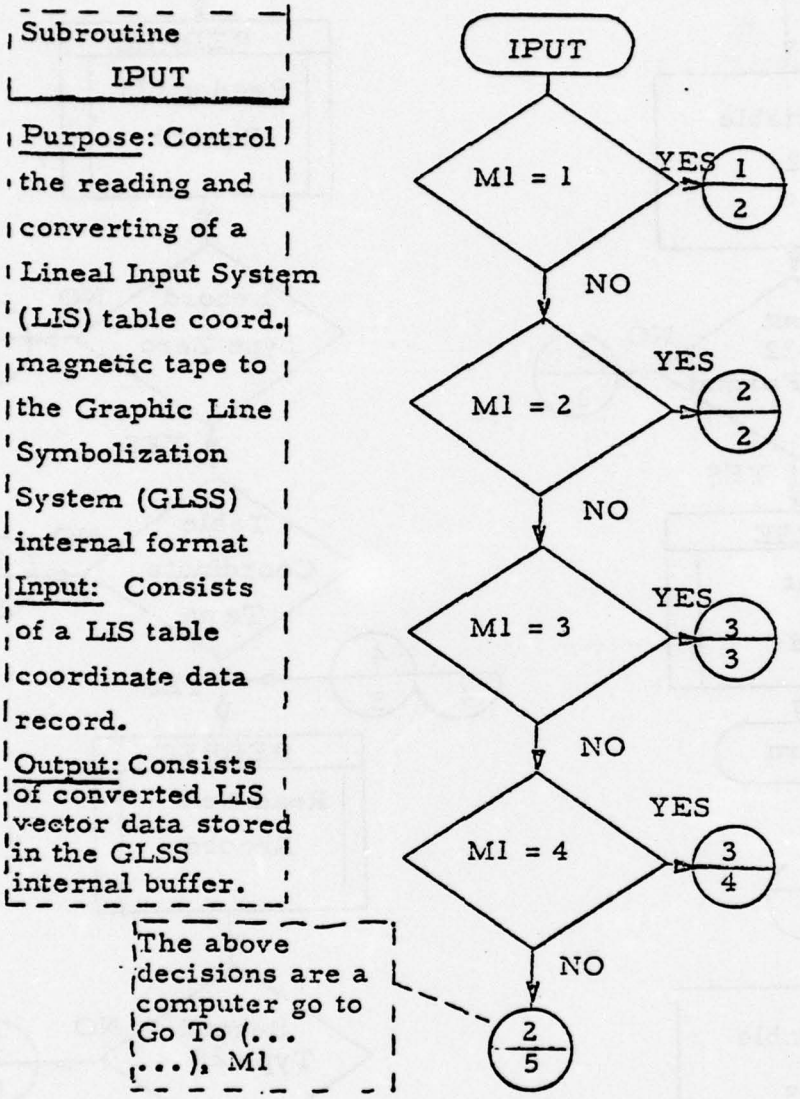



Figure III-3 - IPUT Process Flow

(Page 1 of 5)

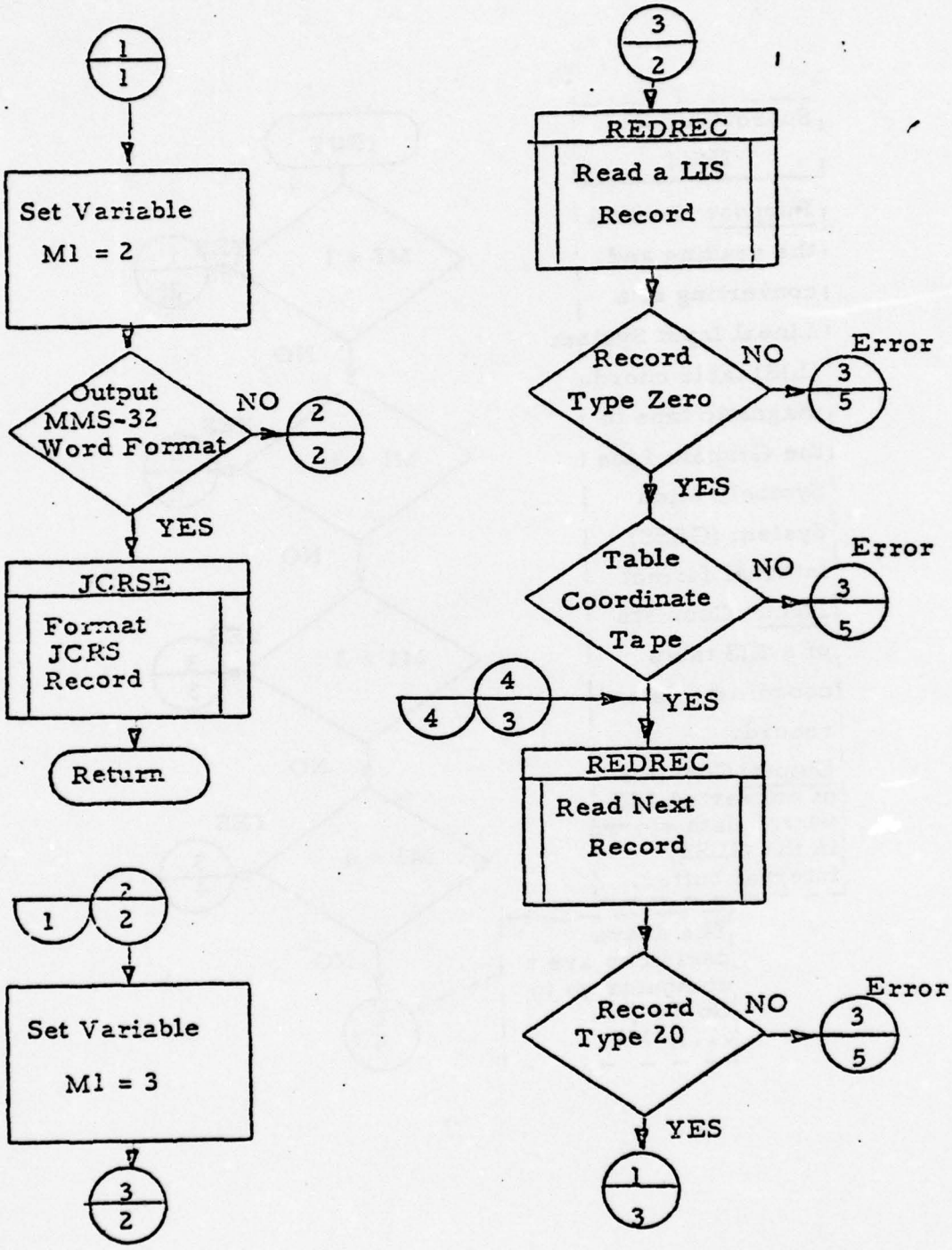



Figure III-3 - IPUT Process Flow (Page 2 of 5)

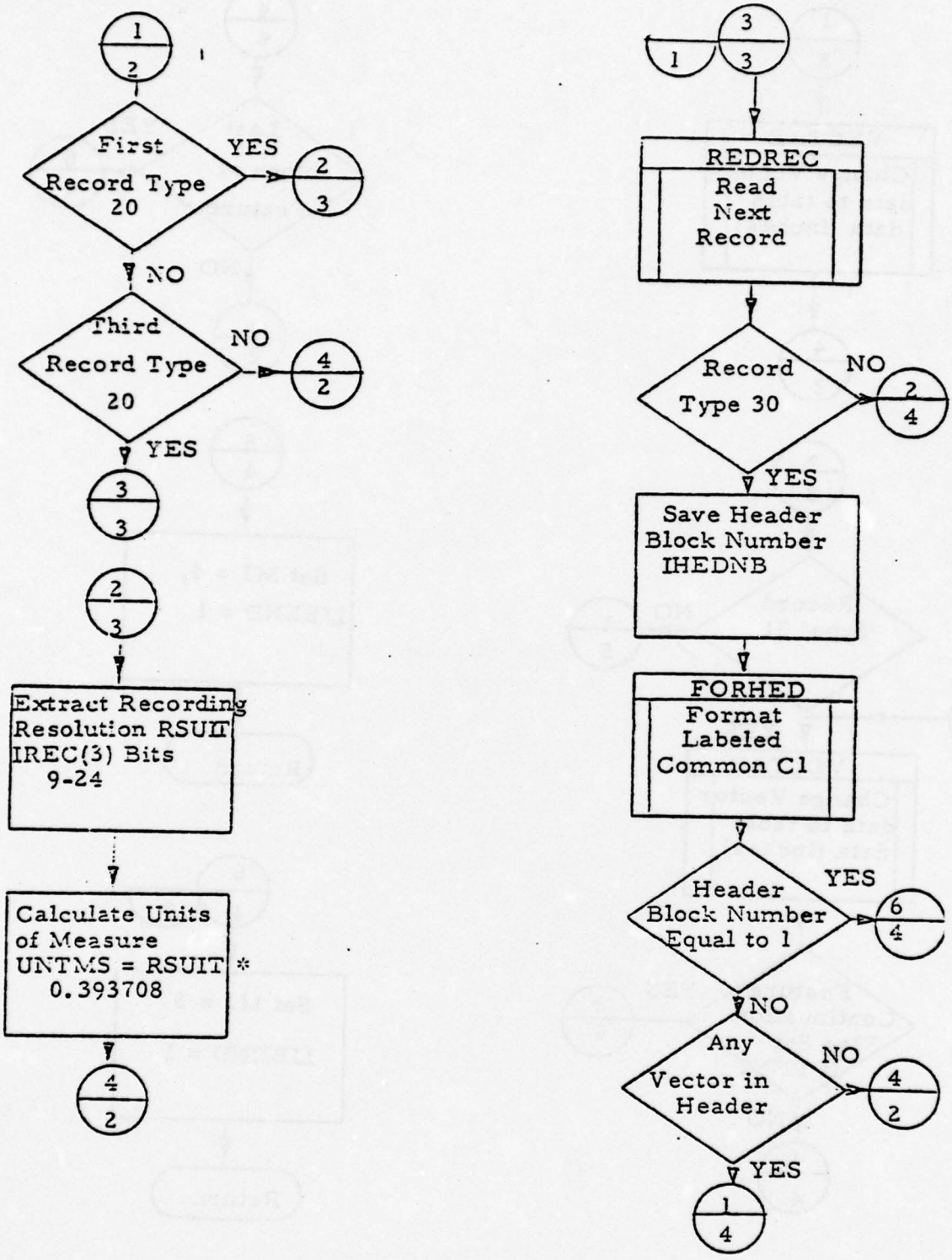



Figure III-3 - IPUT Process Flow

(Page 3 of 5)

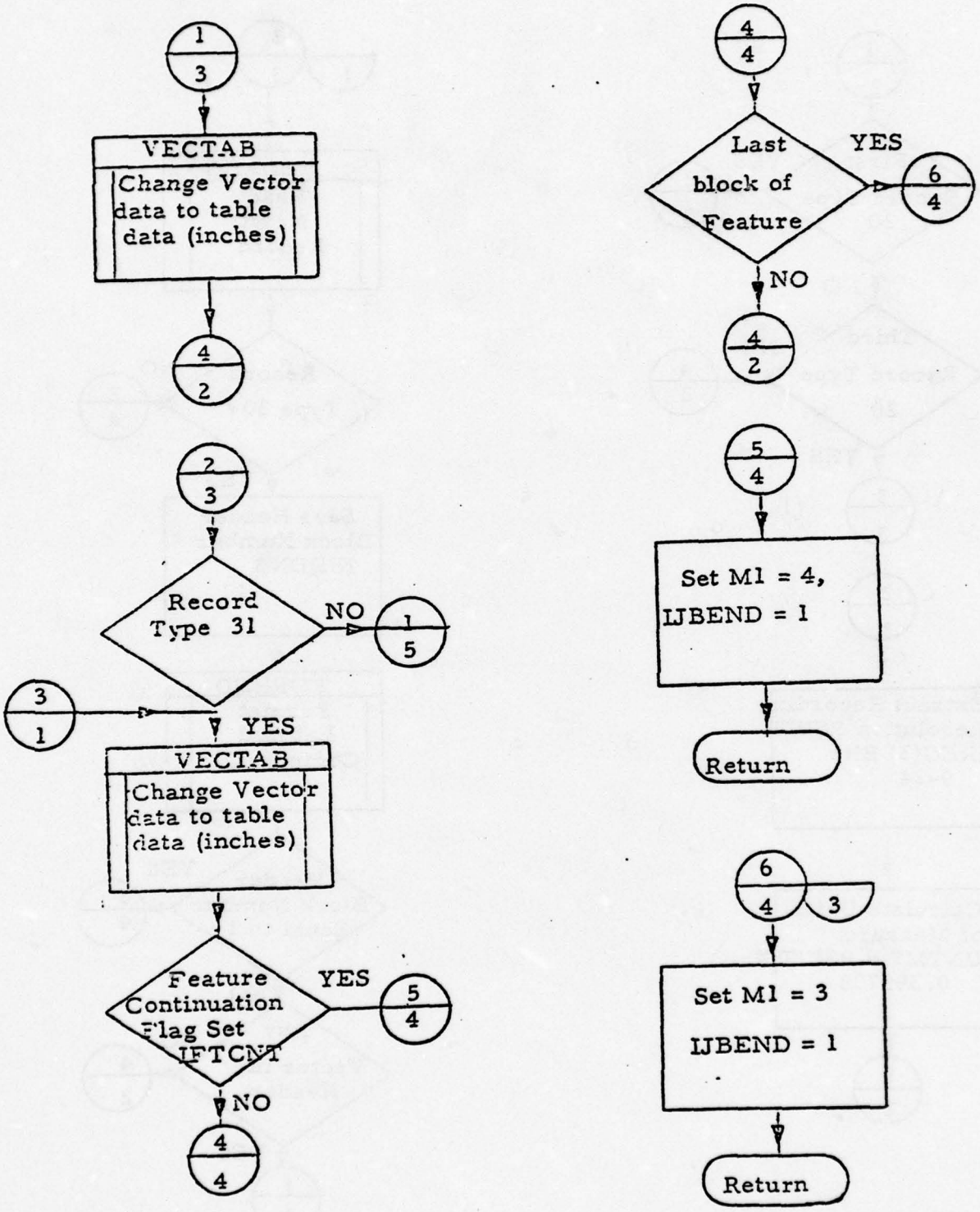



Figure III-3 IPUT Process Flow

(Page 4 of 5)

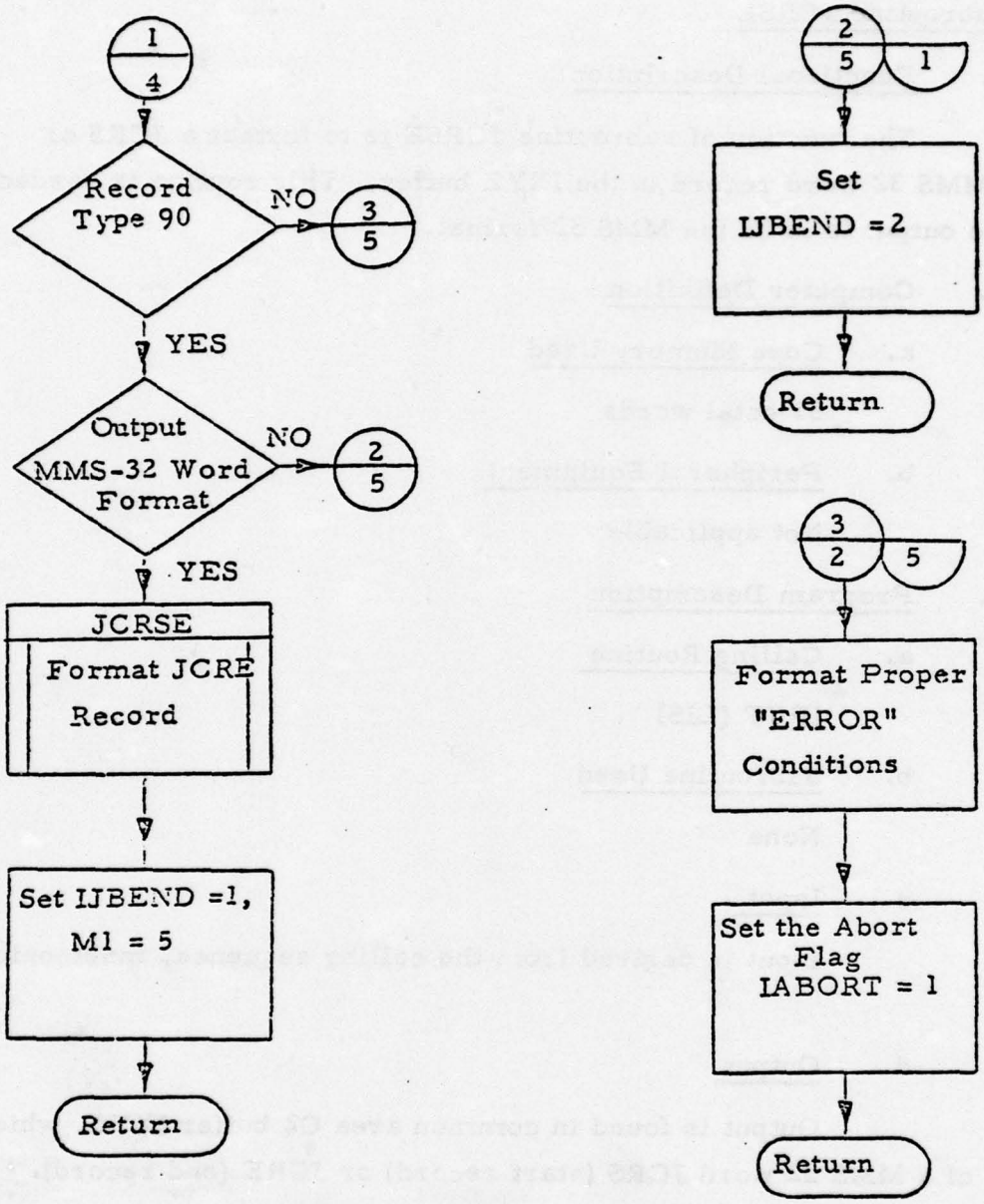



Figure III-3 IPUT Process Flow
(Page 5 of 5)

D. Subroutine JCRSE

1. Functional Description

The function of subroutine JCRSE is to format a JCRS or JCRE MMS 32 word record in the IXYZ buffer. This routine is needed when the output is to be the MMS 32 format.

2. Computer Definition

a. Core Memory Used

51 octal words

b. Peripheral Equipment

Not applicable

3. Program Description

a. Calling Routine

IPUT (LIS)

b. Subroutine Used

None

c. Input

Input is derived from the calling sequence, mnemonic IFOR.

d. Output

Output is found in common area C2 buffer IXYZ, which consist of a MMS 32 word JCRS (start record) or JCRE (end record).

e. Processing Methodology

Processing flow of subroutine JCRSE is shown in Figure III-4. Entry is made via subroutine IPUT with the first

thirty two word of buffer IXYZ being cleared. Mnemonic IFOR is examined and if found to contain a value of one, a MMS 32 word JCRS (start record) is formed in buffer IXYZ with control being returned to calling routine IPUT. If IFOR contains a value of two, a MMS 32 word JCRE (end record) is formed in IXYZ and again process control is returned to the calling routine IPUT. This routine is only called when the output format is the MMS 32 word record type.

f. Calling Sequence

CALL JCRSE (IFOR)
IFOR = 1 form JCRS record
IFOR = 2 form JCRE record

g. Major Algorithm

N/A

h. Program Constants and Variables

IFOR - mnemonic containing start or end record directive.

IXYZ - buffer containing (first thirty two words) the JCRS or JCRE record.

5. Error Conditions

None

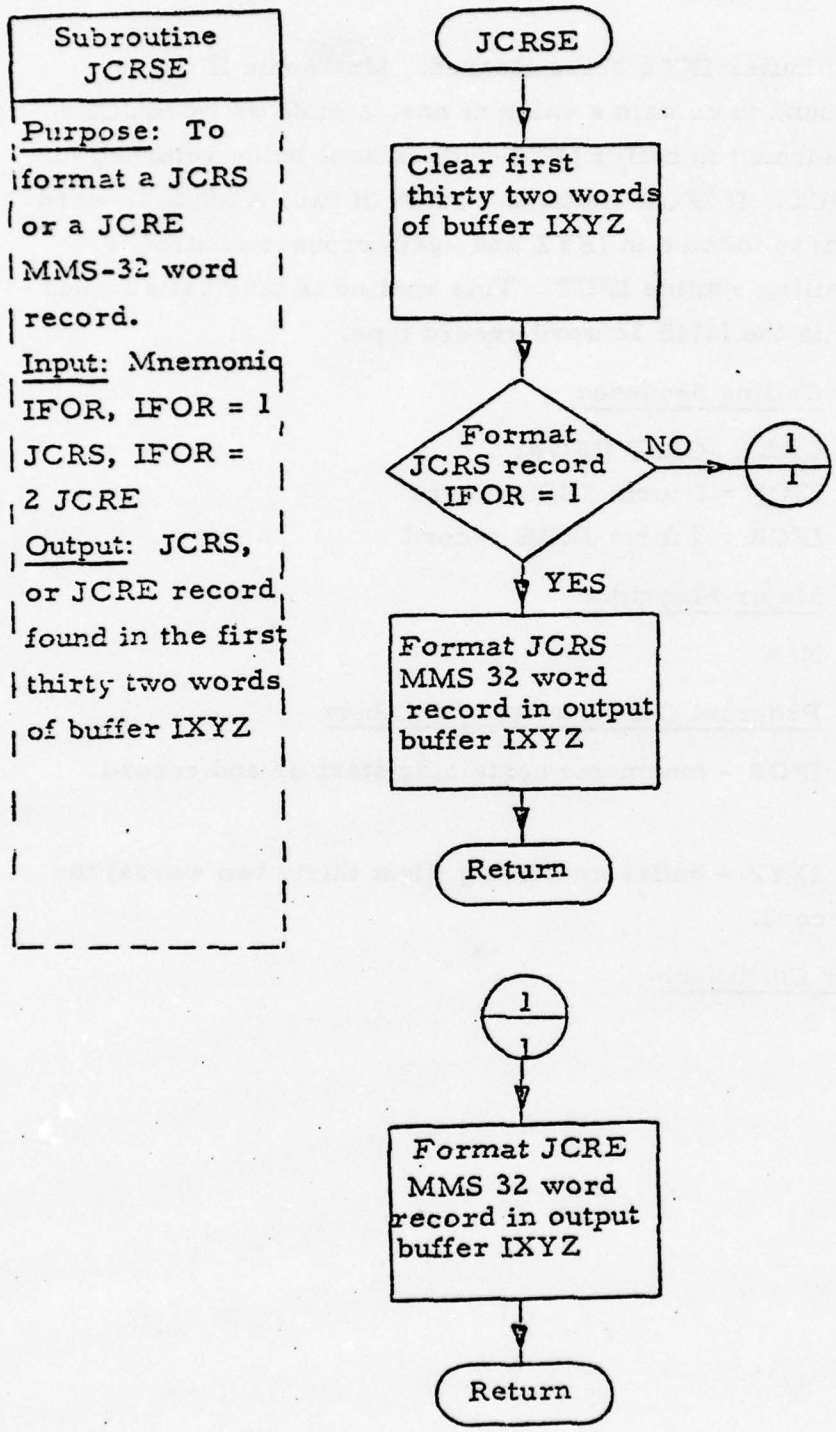



Figure III-4 JCRSE Process Flow

(Page 1 of 1)

E. Subroutine VECTAB

1. Functional Description

The function of subroutine VECTAB is to convert a LIS vector data list to absolute X, Y table coordinate values and to store these values into the GLSS internal coordinate buffer.

2. Computer Definition

a. Core Memory Used

513 octal words

b. Peripheral Equipment

N/A

3. Program Description

a. Calling Routine

IPUT (LIS)

b. Subroutine Used

N/A

c. Input

Input consists of data found in common area LIS, namely mnemonics IREC (buffer containing record type 30 or 31), UNTMS (recording resolution converted to inches), ISVPN (starting vector position number) and NUMVEC (number of vectors in block). Other input consists of mnemonic IHEDIN (header number input) and ISMOTH (smooth option).

d. Output

The primary output consists of data found in common area C2, feature line center data, namely buffer IXYS which

contains the LIS vector data converted to absolute X, Y table coordinates in inches. Other output found in common C2, is the number of data points in buffer IXYZ mnemonic NUMPTS sub-current index pointer ICURDX. The feature continuation flag IFTCNT is also used in subroutine VECTAB to indicate that buffer IXYZ has reached its upper limit and more data exists for the feature in question.

e. Processing Methodology

The processing flow of subroutine VECTAB is shown in Figure III-5. Process control is made via subroutine IPUT when vector data is found in record types 30 or 31. Upon first entry the incremental values of the vector code, array (VEC) are initialized to the units of measure, that is the incremental values times the units of measure (inch) mnemonic UNTMS. UNTMS is the recording resolution (microns) times the microns to inches conversion value. On subsequent entries or after array VEC has been initialized, a new feature header input check is made. If a new header has been input, needed flags and pointers are reset. If subroutine VECTAB call-back flag (IVECLK) and feature continuation flag (IFTCNT) are not set, the start stop limits for storing coordinates values are calculated utilizing the smooth option input (ISMOTH). The starting coordinates (absolute X, Y values) are placed in mnemonics XTEMP and YTEMP, respectively. The index within array IREC pointer (mnemonic INDX) and vector index within word pointer (IVECDX) are then calculated. The vector code is extracted from buffer IREC (see Figure III-6) utilizing pointer INDX and IVECDX. This code is then placed in mnemonic IVDX and is used as a pointer into array VEC. The delta coordinates then are summed to XTEMP and YTEMP and stored into output buffer IXYZ. The number of points input (NPTSIN) and the number of points placed in the output buffer (NUMPTS) are incremented. If the vector processed count (ICTVEC) is equal to the number of vectors input (NUMVEC) the VECTAB call-back flag (IVECLK) is set and control returned to the

calling routine IPUT. If the output buffer's index reaches its upper limit (i.e., buffer is full) the feature continuation flag (IFTCNT) is set and processing control returned to the calling routine IPUT.

Upon subsequent entries, when the VECTAB call-back flag (IVECLK) is found set, process control is passed to the above mentioned index within array (INDX) and index within word (IVECDX) calculations with the aforementioned processing being continued. If IVECLK is not set and the feature continuation flag (IFTCNT) is set, new start stop limits for storing coordinate values are calculated, again using the smooth option input (ISMOTH). Certain flags and pointers are then initialized and process control is passed to the above mentioned vector code extraction processing area.

f. Calling Sequences

Call VECTAB

g. Major Algorithms

None

4. Program Constants and Variables

IREC - array containing record type 30 or 31.

UNTMS - units of measure (recording resolution changed to inches).

ISVPN - starting vector position number.

NUMVEC - number of vector within IREC.

IHEDIN - number of headers input.

ISMOTH - smooth data option.

IXYZ - buffer containing output coordinates.

NUMPTS (ICURDX) - number of points in above buffer.

ICURDX - current buffer pointer (1-5)

IFTCNT - feature continuation flag.

VEC - array containing L.I.S. delta X, Y increments.

IVECLK - subroutine VECTAB call-back flag.

XTEMP - summed X temporary location.

XTEMP - summed Y temporary location.

INDX - index pointer within buffer IREC.

IVECDX - vector within word index.

IVDX - L.I.S. four bit vector (right justified).

NPTSIN - number of points input.

ICTVEC - vector processed count.

5. Error Conditions

None

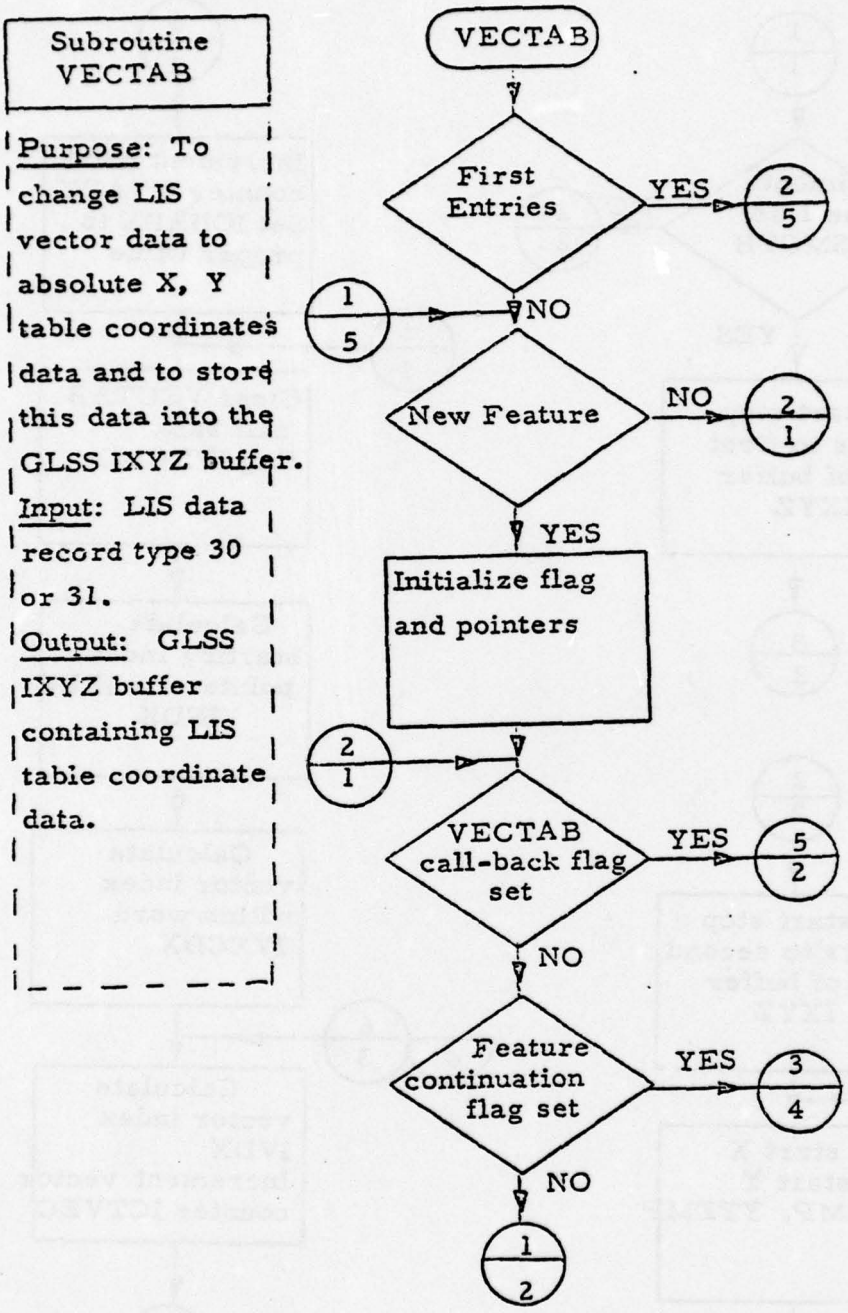



Figure III-5 VECTAB Process Flow (Page 1 of 5)

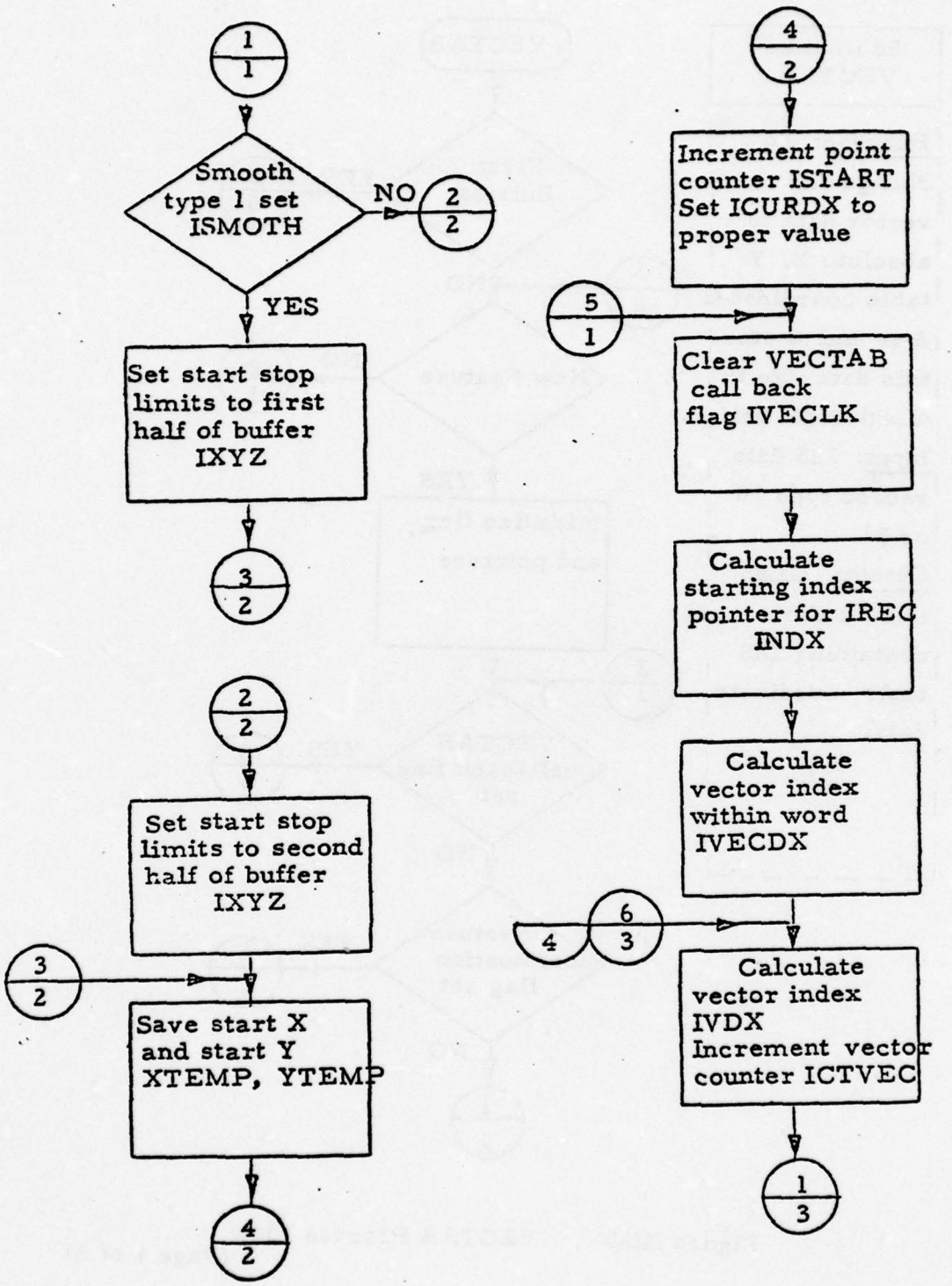



Figure III-5 VECTAB Process Flow

(Page 2 of 5)

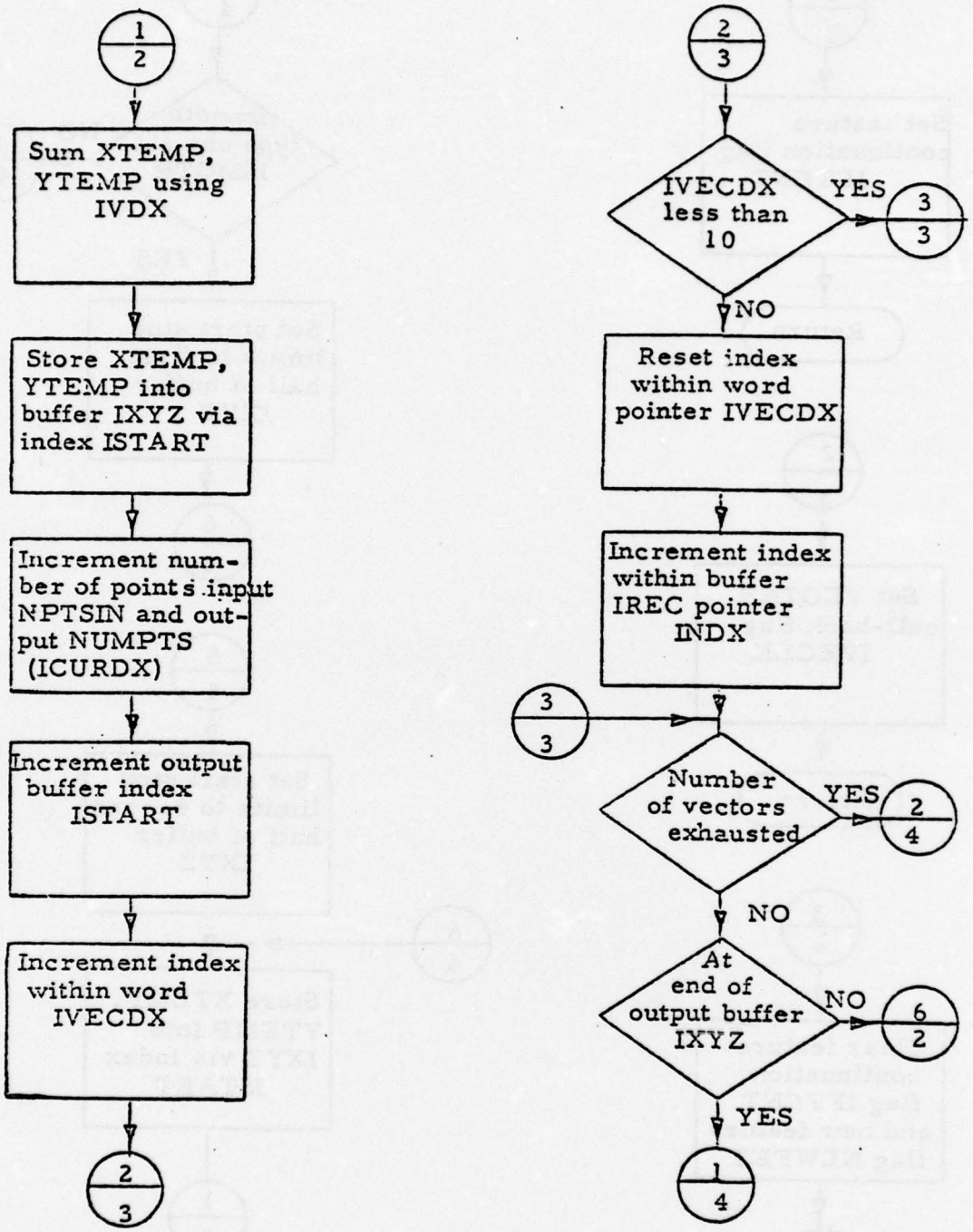



Figure III-5 VECTAB Process Flow
(Page 3 of 5)

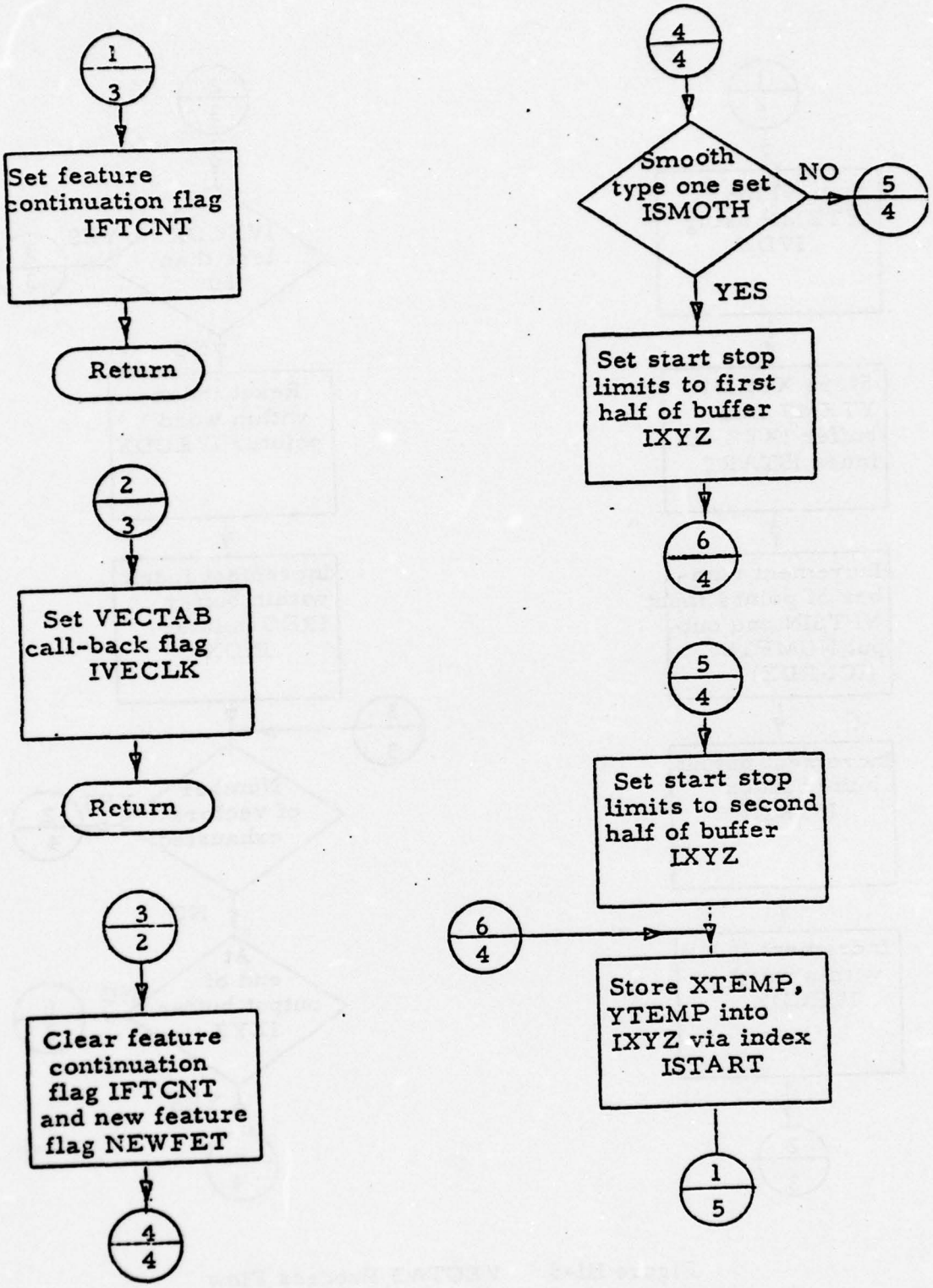



Figure III-5 VECTAB Process Flow (Page 4 of 5)

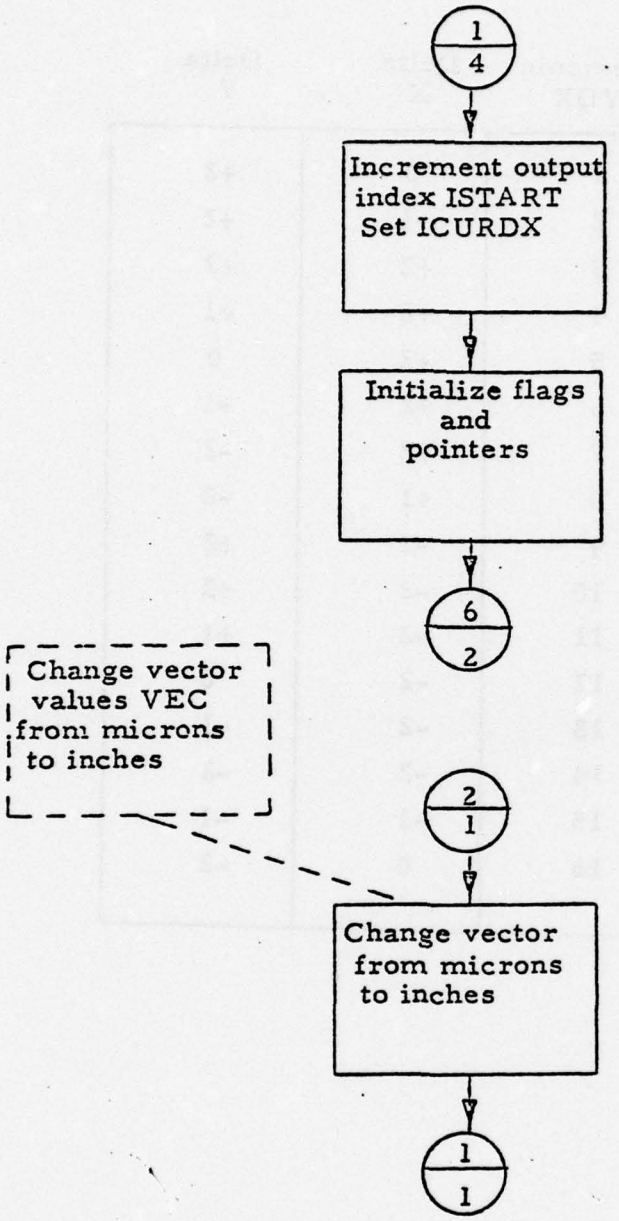



Figure III-5 VECTAB Process Flow
(Page 5 of 5)

| Numeric Code | Mnemonic IVDX | Delta X | Delta Y |
|---|---|---|---|
| 0 | 1 | 0 | +2 |
| 1 | 2 | +1 | +2 |
| 2 | 3 | +2 | +2 |
| 3 | 4 | +2 | +1 |
| 4 | 5 | +2 | 0 |
| 5 | 6 | +2 | -1 |
| 6 | 7 | +2 | -2 |
| 7 | 8 | +1 | -2 |
| 8 | 9 | -1 | +2 |
| 9 | 10 | -2 | +2 |
| 10 | 11 | -2 | +1 |
| 11 | 12 | -2 | 0 |
| 12 | 13 | -2 | -1 |
| 13 | 14 | -2 | -2 |
| 14 | 15 | -1 | -1 |
| 15 | 16 | 0 | -2 |

Figure III-6 - LIS Vector Code and Incremental Values
(Page 1 of 1)

## F. FORHED

### 1. Functional Description

The primary function of subroutine FORHED is to extract from a Lineal Input System's header record feature data and to reformat this data for the Graphic Line Symbolization System's internal buffer.

### 2. Computer Definition

#### a. Core Memory Used

753 octal words

#### b. Peripheral Equipment

N/A

### 3. Program Description

#### a. Calling Routine

IPUT

#### b. Subroutine Used

N/A

#### c. Input

Input consists of a L.I.S. header record, record type 30, located in labeled common area LIS mnemonic IREC (buffer containing input record).

#### d. Output

Output from subroutine FORHED is found in labeled common areas C1 and C2. The output consists of feature class, type and subtype (mnemonic ICLSS1), eight codified descriptors (mnemonics ICLSS2 and ICLSS3), bounding rectangle information (mnemonics IXMIN, IYMIN, IXMAX, IYMAY), first, last coordinate point (mnemonics IXFST, IYFST, IXLST, IYLST) and special numerics (mnemonic IHEAD(1)).

Other output consists of the first coordinate point, converted to inches, store in buffer IXYZ.

e. Processing Methodology

Processing flow of subroutine FORHED is depicted in Figure III-7. Entry is made via subroutine IPUT with GLSS system flags and pointers being initialized. The current index pointer (mnemonic ICURDX) is initialized with the number of header record and data record input being incremented (mnemonic IHEDIN and IDTIN respectively). The feature class, type, subtype and eight codified descriptors are extracted from input buffer IREC converted to Fieldata and stored in mnemonics ICLSS1, ICLSS2, and ICLSS3, respectively. The special numerics, stored in ASCII, are extracted from IREC, converted to Fieldata and stored in mnemonic IHEAD(1). The first X and Y coordinate point (microns) is removed from IREC, converted to inches and stored in mnemonic IXFST and IYFST. They are also stored in feature output buffer IXYZ directed by ICURDX. The last coordinate point and the bounding rectangle information are then removed from IREC, converted to inches and stored in their respective areas found in labeled common C1. Process control is then returned to the calling routine IPUT.

f. Calling Sequence

N/A

g. Major Algorithms

N/A

4. Program Constants and Variables

Labeled common area C1, mnemonics:

ICLSS1 - feature, class, type, sub-type (Fieldata)
ICLSS2 - six codified descriptors
ICLSS3 - two codified descriptors
IXMIN - minimum X

IYMIN - minimum Y

IXMAX - maximum X

IYMAX - maximum Y

IXFST - first X of feature data list

IYFST - first Y of feature data list

IXLST - last X of feature data list

IYLST - last Y of feature data list

IHEAD - buffer containing 16 text word

5. Error Conditions

N/A

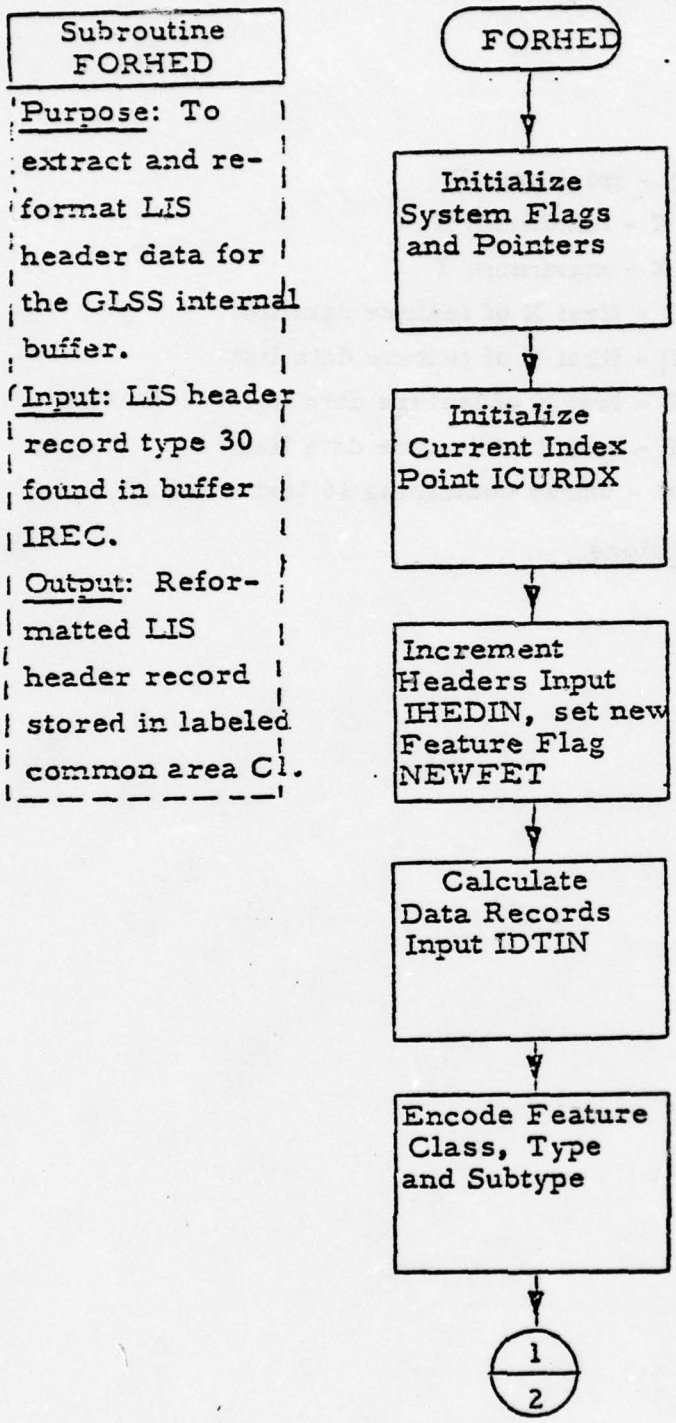



Figure III-7 FORHED Process Flow (Page 1 of 3)

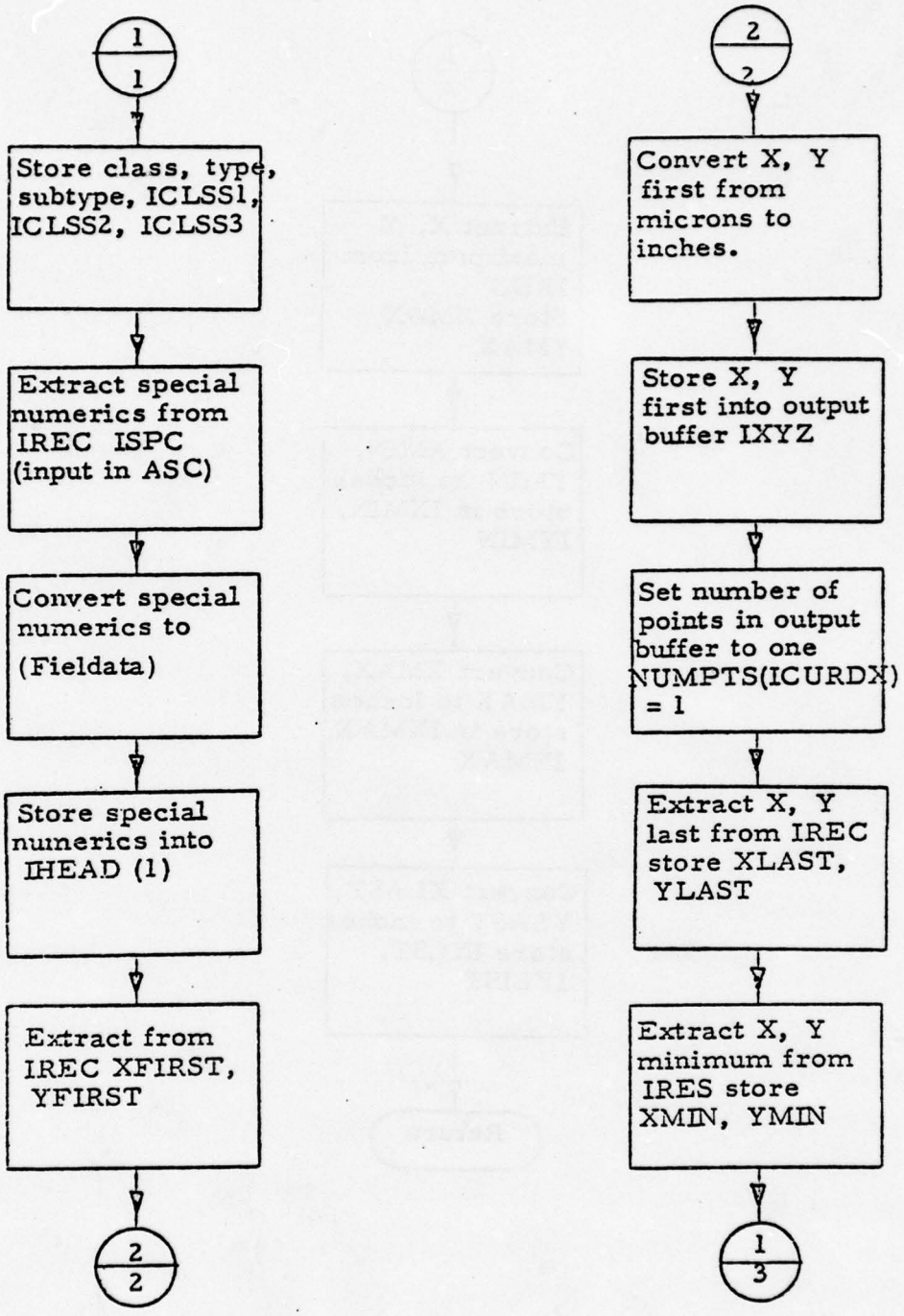



Figure III-7 FORHED Process Flow (Page 2 of 3)

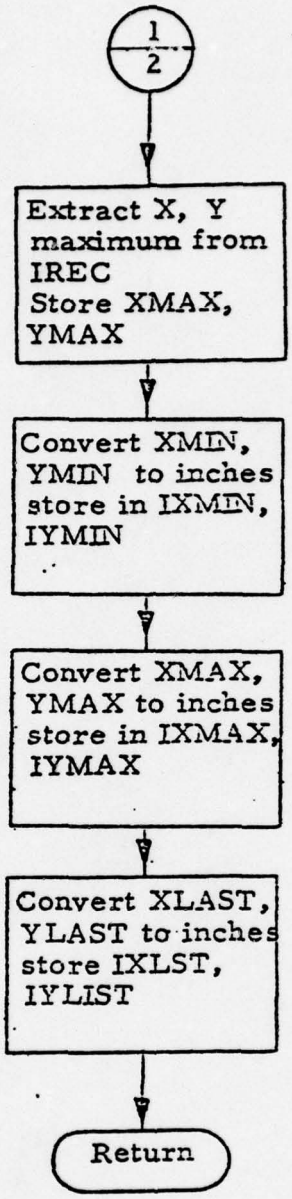



Figure III- 7 FORHED Process Flow (Page 3 of 3)

## G. Subroutine REDREC

### 1. Functional Description

The function of REDREC is to read a Lineal Input System (LIS) table coordinate magnetic tape data list. A secondary function is to strip from a data record common data block attributes.

### 2. Computer Definition

#### a. Core Memory Used

202 octal words.

#### b. Peripheral Equipment

Input, 9-track magnetic tape non system standard.

### 3. Program Description

#### a. Calling Routine

IPUT (LIS)

#### b. Subroutines Used

None

#### c. Input

Input consists of a LIS 9-track magnetic tape containing table coordinate data. Appendix I depicts the file layout and format of the LIS data records.

#### d. Output

Output from subroutine REDREC is found in common area LIS. The output consists of the buffer containing a LIS record (mnemonic IREC), record type (mnemonic IRTYPE), data block number (IBLNUM), starting vector position number (ISVPN), number of vectors in block (NUMVEC) and feature number (IFETNO).

e. Processing Methodology

Processing methodology of subroutine REDREC is depicted in flow diagram, Figure III-8. Entry is made via subroutine IPUT. If first entry, the LIS header record is read and the first end of file is sensed. On subsequent entries or the above first end of file, the next data record is read. The record type (IRTYPE), starting vector position number (ISVPN), number of data vectors (NUMVEC), data block number (IBLNUM), and feature number (IFETNO) are unpacked from their respective areas within buffer IREC. Control is then returned to the calling routine IPUT. If a second end of files is reached, mnemonic IENDTP is set and control is returned to IPUT.

f. Calling Sequence

Call REDREC

g. Major Algorithms

None

4. Program Constants and Variables

IREC - buffer containing LIS data record.
IRTYPE - LIS record type.
ISVPN - starting vector position number.
NUMVEC - number of vectors in data block.
IBLNUM - data block number.
IFETNO - feature number.
IENDTP - end of data flag.

5. Error Condition

o "TAPE ERROR"

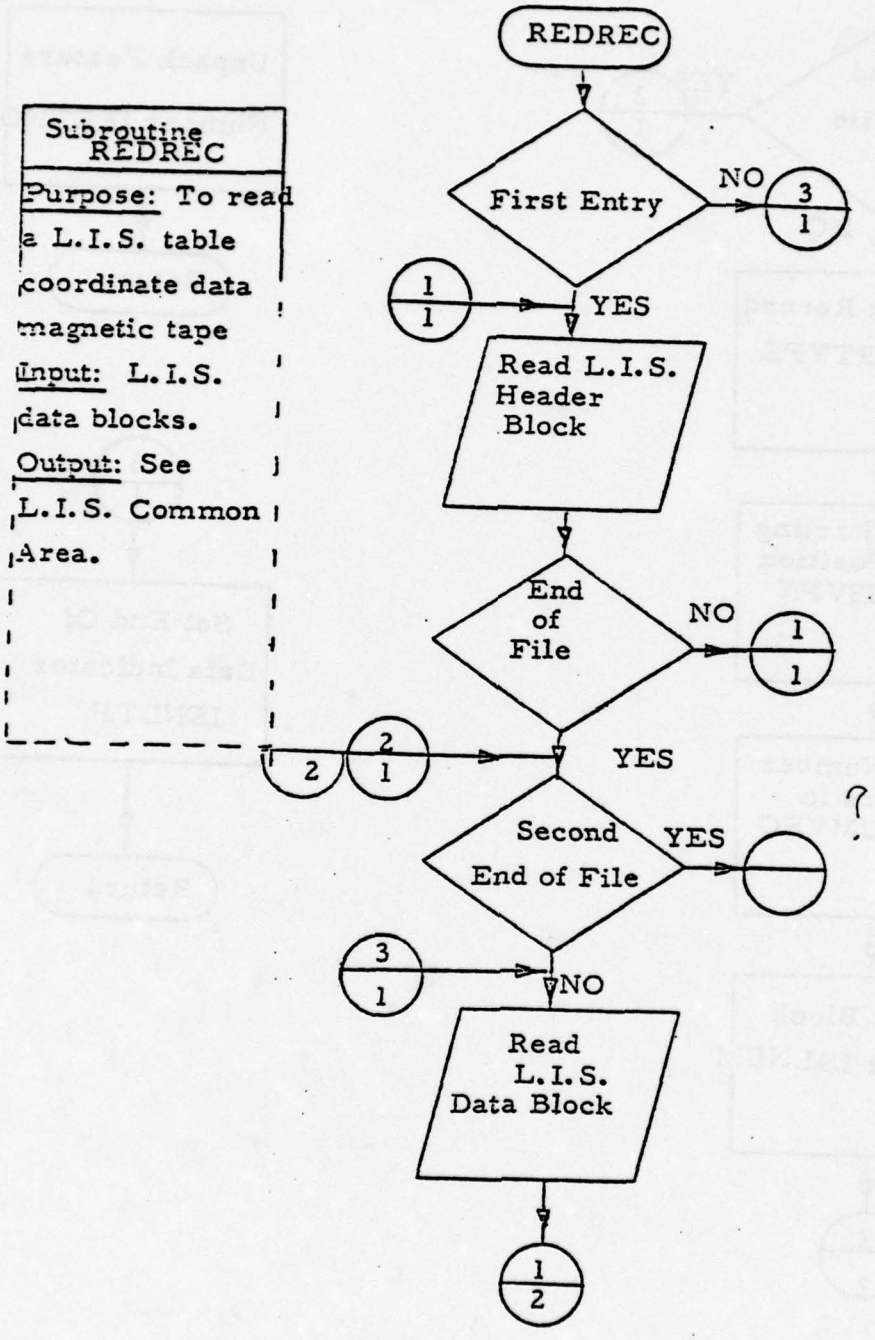



Figure III-8 - REDREC Process Flow

(Page 1 of 2)

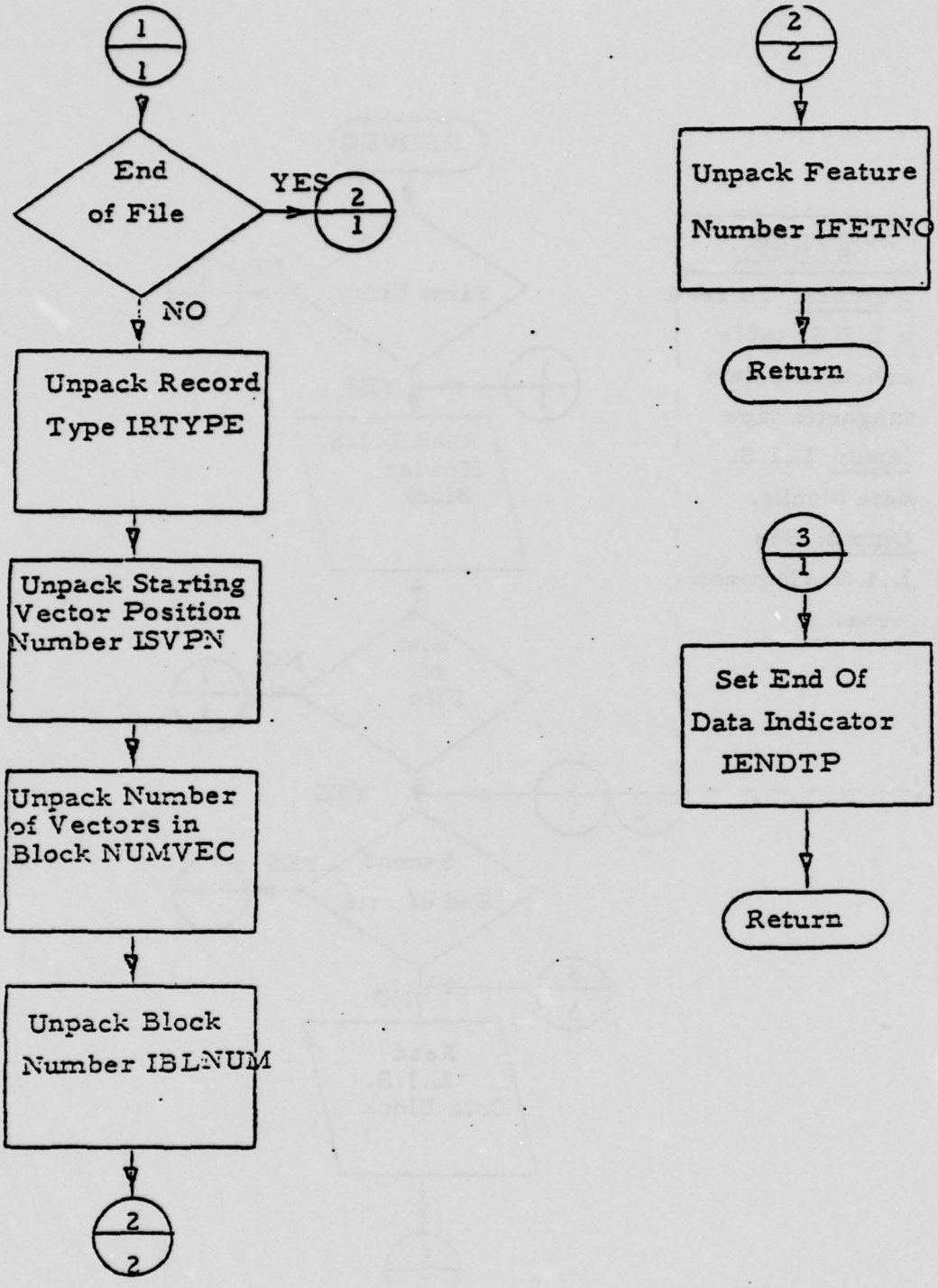



Figure III-8 - REDREC Process Flow

(Page 2 of 2)

## H. OPUT (Gerber 2032 Plotter)

### 1. Functional Description

Subroutine OPUT (Gerber 2032 Plotter Output) primary processing function is to produce a Gerber Plotter magnetic tape by utilizing the plotter subroutines described in the Gerber Plotter Subroutines Manual for Users, Unique Number USL-043, May 1973.

### 2. Computer Definition

#### a. Core Memory Used

4501 octal words.

#### b. Peripheral Equipment

Nine track magnetic tape unit.

### 3. Program Description

#### a. Calling Routine

MONITR

#### b. Subroutines Used

The following Gerber plotter subroutines are used to generate drafting instructions:

| | |
|---|---|
| INIT | LINES |
| OTYPE | FLASH |
| MOVE | MESAGE |
| SEL | DONE |

#### c. Input

The input to subroutine OPUT is obtained from blank common, namely output coordinate data, buffer mnemonic IXYZ, the current buffer index pointer mnemonic ICURDX and the number of points to output mnemonic NUMPTS (ICURDX). Other input consists of the symbol piece line weight mnemonic ISYPLW.

d. Output

Output consists of a nine track magnetic tape containing Gerber plotter drafting instructions generated by the interaction of the Gerber Plotter Subroutines mentioned above.

e. Processing Methodology

Entry to Subroutine OPUT is made from GLSS controller MONITR. Figure 9 depicts the process flow of OPUT. On first entry, Gerber plotter initialization subroutine INIT is called to set constants, variables and output file code (09) needed by the Gerber subroutine. If the user requested the print option IOFLAG W set = 4. Subroutine OTYPE is then called to set the output device code, usually magnetic tape. Upon subsequent entries, process control is passed to the end of process check. If mnemonic IJBEND is not equal to two (normal process) output report counters are incremented and the starting output index pointer is generated. The number of points to output is then extracted (NPTS) and the coordinate data is scaled with the resultant X, Y's placed in their respective output buffers (SGER and YGER). Gerber Subroutine MOVE is called to position the photo head at the first coordinate value of the output buffers. The symbol piece type line weight (mnemonic ISYPLW (ISYDEX)) is placed in mnemonic IGERSZ. The aperture setting for the photo head is then found utilizing the symbol piece line weight (IGERSZ). Gerber Subroutine SEL is called to select the aperture setting generated above. If the number of points to output (NPTS) is greater than one, Gerber Subroutine LINE is called with the appropriate arguments with process control being returned to the calling routine MONITR. If the number of points to output is one (dot symbol) Gerber Subroutine FLASH is called to generate a command to flash the current aperture selected by the prior call to SEL. Process control is again returned to the calling routine. When an end of process indication is found (mnemonic IJBEND is equal to two) Gerber Subroutine MESAGE is called causing an output message to be generated

(on magnetic tape) for the Gerber plotter operator indicating end of plot. The last Gerber Subroutine to be called is DONE which causes the plotting head to be moved to the lower center of the plotting table and the plotter to halt. Control is then returned to MONITR.

f. Calling Sequence

Call OPUT

g. Major Algorithms

None

4. Program Constants and Variables

NPTS - number of coordinate points to output.

XGER - 1000 word buffer containing X values.

YGER - 1000 word buffer containing Y values.

IAPNB - current aperture selection number.

5. Error Conditions

None

**Subroutine OPUT**

(Gerber Output Routine)

Purpose - To generate a Gerber plot tape via interaction with GERBER Subroutines.

Input- Data Found in blank common namely IXYZ buffer, ICURDX and NUMPTS (ICURDX)

Output - Nine track magnetic tape containing Gerber Plotter drafting instructions.

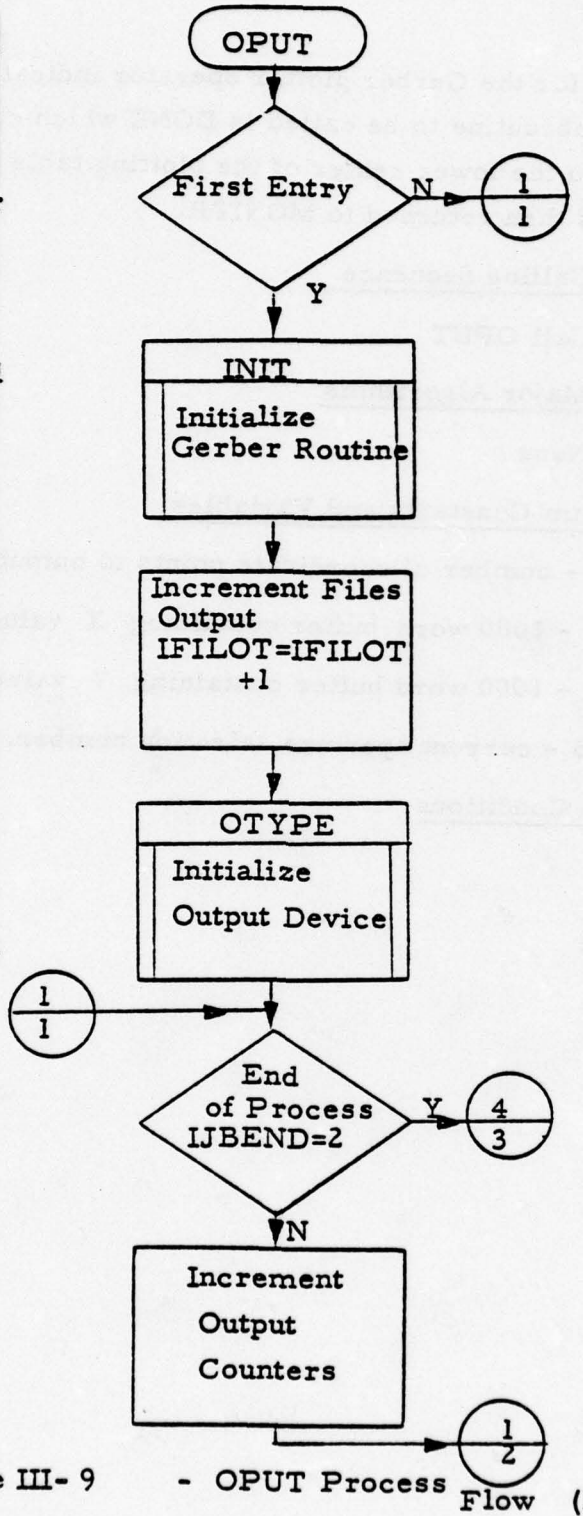



Figure III- 9 - OPUT Process Flow (Page 1 of 3)

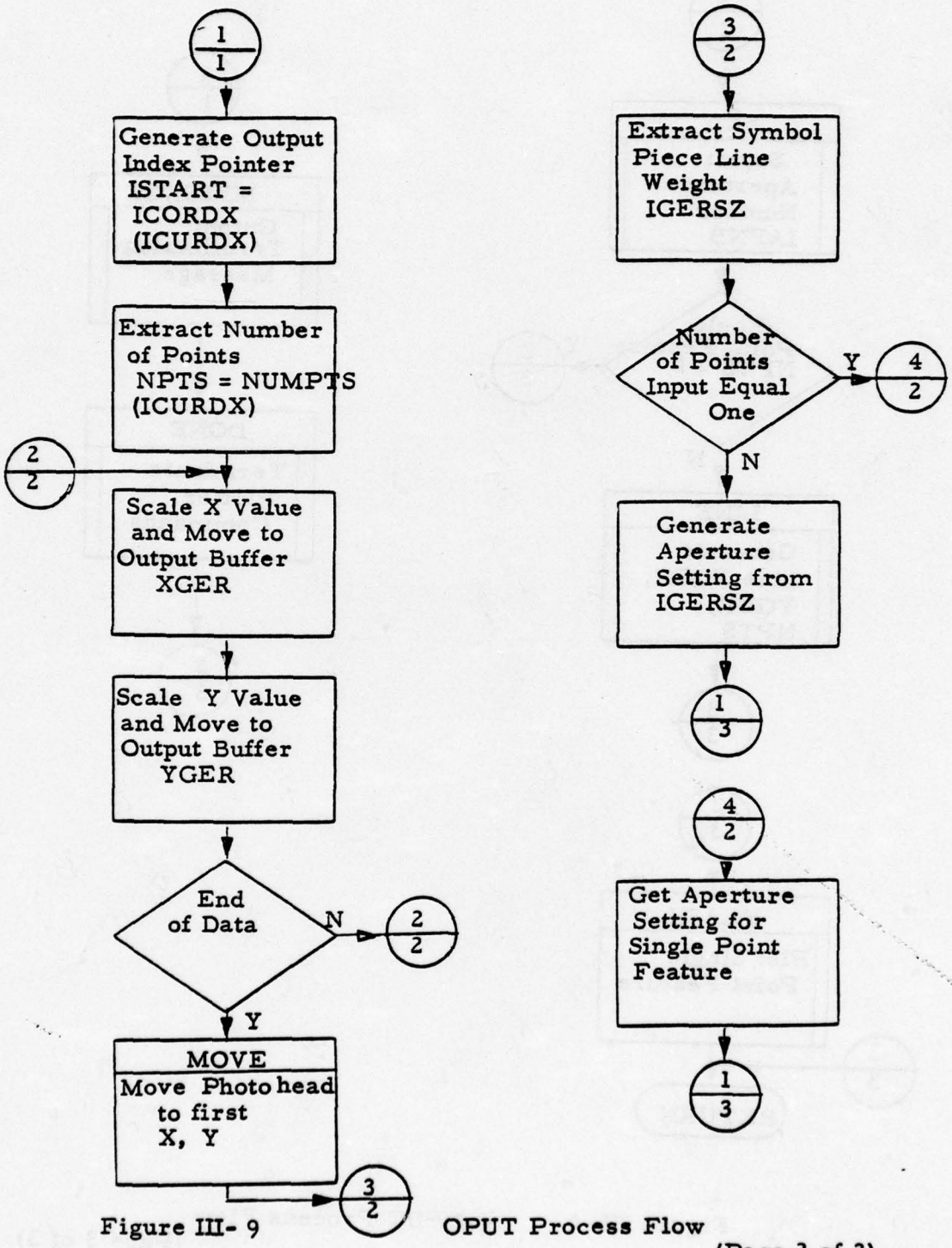



Figure III- 9 OPUT Process Flow

(Page 2 of 3)

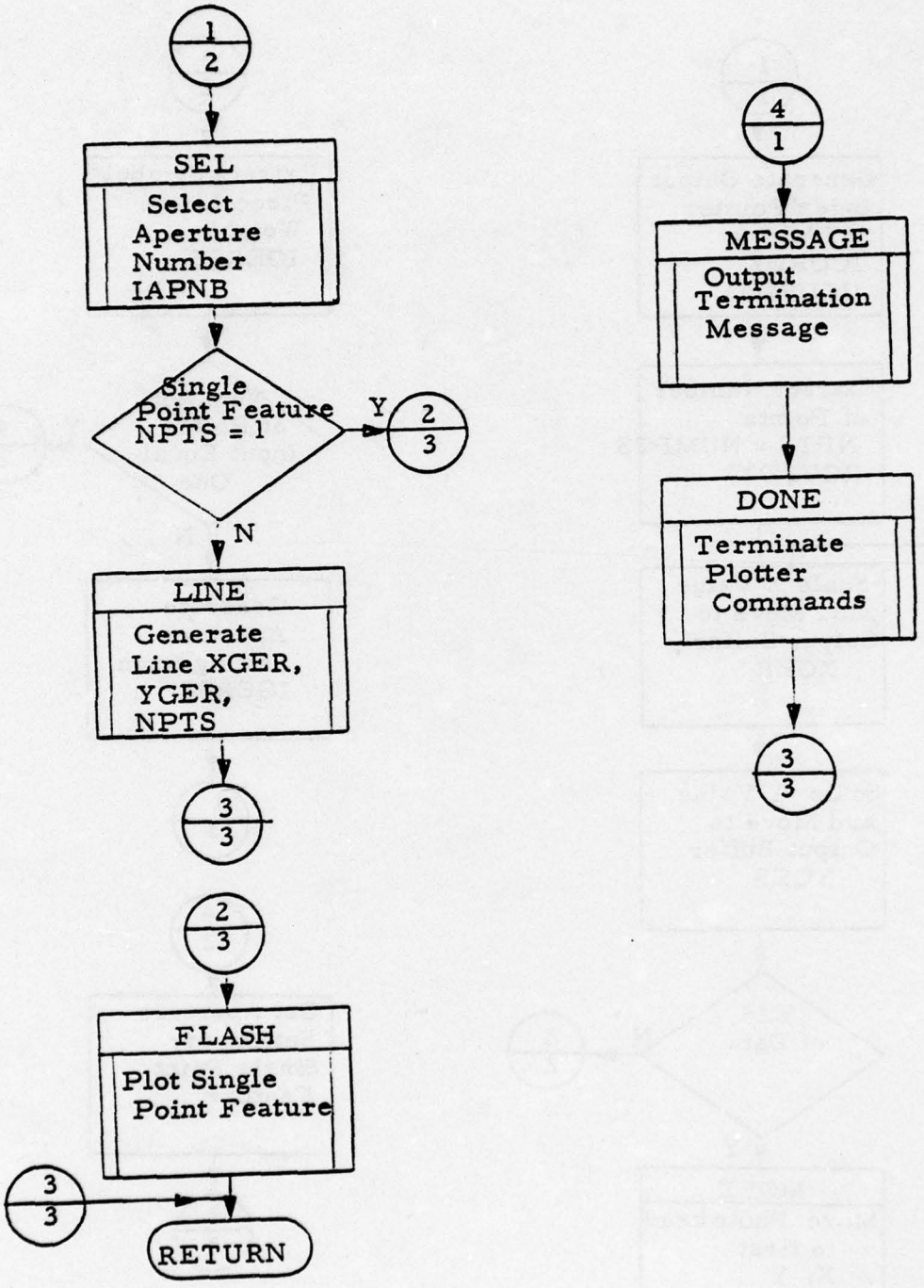



Figure III-9 - OPUT Process Flow (Page 3 of 3)

## I. OPUT (XYNETICS Plotter)

### 1. Functional Description

Subroutine OPUT (XYNETICS Plotter Output) primary processing function is to produce a XYNETICS plotter magnetic tape by utilizing the XYNETICS Standard User FORTRAN package.

### 2. Computer Definition

#### a. Core Memory Used

232 octal words.

#### b. Peripheral Equipment

Nine track magnetic tape unit.

### 3. Program Description

#### a. Calling Routine

MONITR

#### b. Subroutines Used

The following XYNETICS plotter subroutines are used to generate plotter directives:

DIMTAB
PLOTX

#### c. Input

The input to subroutine OPUT is obtained from blank common, namely output coordinate data, buffer mnemonic IXYZ, the current buffer index pointer mnemonic ICURDX and the number of points to output NUMPTS (ICURDX).

#### d. Output

Output consists of a nine track magnetic tape containing

XYNETICS plotter directives generated by the interaction of the XYNETICS subroutines mentioned above.

c. Processing Methodology

Entry to Subroutine OPUT is attained from GLSS controller MONITR. Figure III-10 depicts the processing methodology of OPUT. On first entry, XYNETICS plotter initialization Subroutine DIMTAB is called to establish the drawing surface and logical output device code number. If the input data is LIS, the scale factor, mnemonic SCLFAK, is set to metric (centimeters times ten thousand), otherwise it is left in English. Upon subsequent entries process control is passed to the end of job interrogation. If mnemonic IJBEND is not equal to two, output report counters are incremented, starting output index pointer generated and the number of points to output extracted (NPTS). The first coordinate point, of the symbol piece in question, is extracted and scaled with XYNETICS Subroutine PLOTX being called to move to the start of the feature with the pen up. The symbol piece coordinate data is then scaled and subroutine PLOTX is repeatedly executed, with the pen down command, until the end of the data list is reached. PLOTX is then called to issue a pen up command at the last data point output with process control being returned to the calling routine MONITR. Upon encountering an end of job indication, mnemonic IJBEND equal to two, subroutine PLOTX is called to issue an end of plot command. Process control is returned to the calling routine MONITR.

f. Calling Sequence

Call OPUT

g. Major Algorithms

None

4. Program Constants and Variables

NPTS - number of coordinate points to output.

SCLFAK - scale factor to be applied to coordinate date,
i.e., if data stored in metric SCLFAK = 25400;
if data stored in English SCLFAK = 10000.

IJBEND - end of job indicator.

5. Error Conditions

None

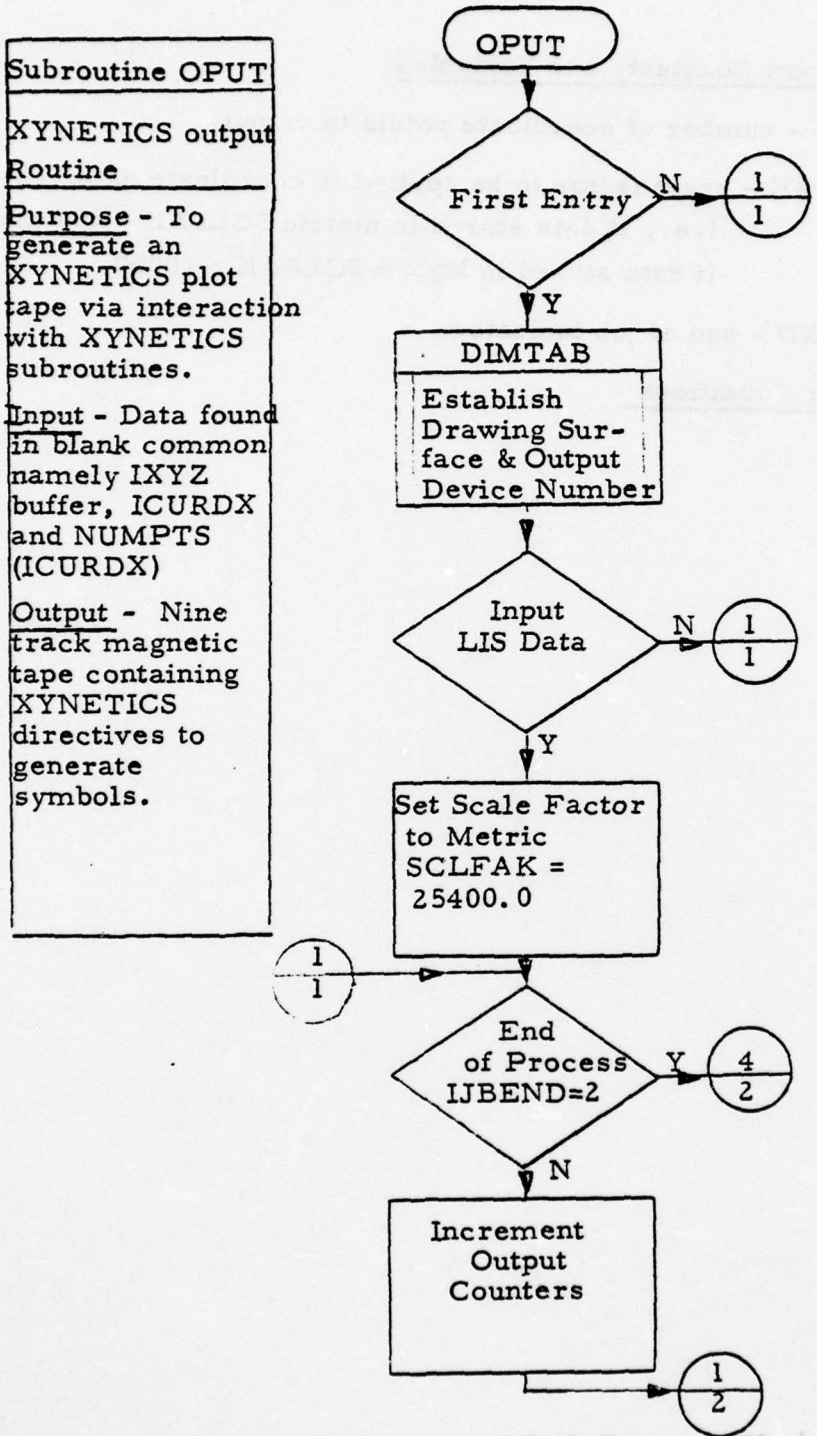



Figure III-10 - OPUT Process Flow (Page 1 of 2)

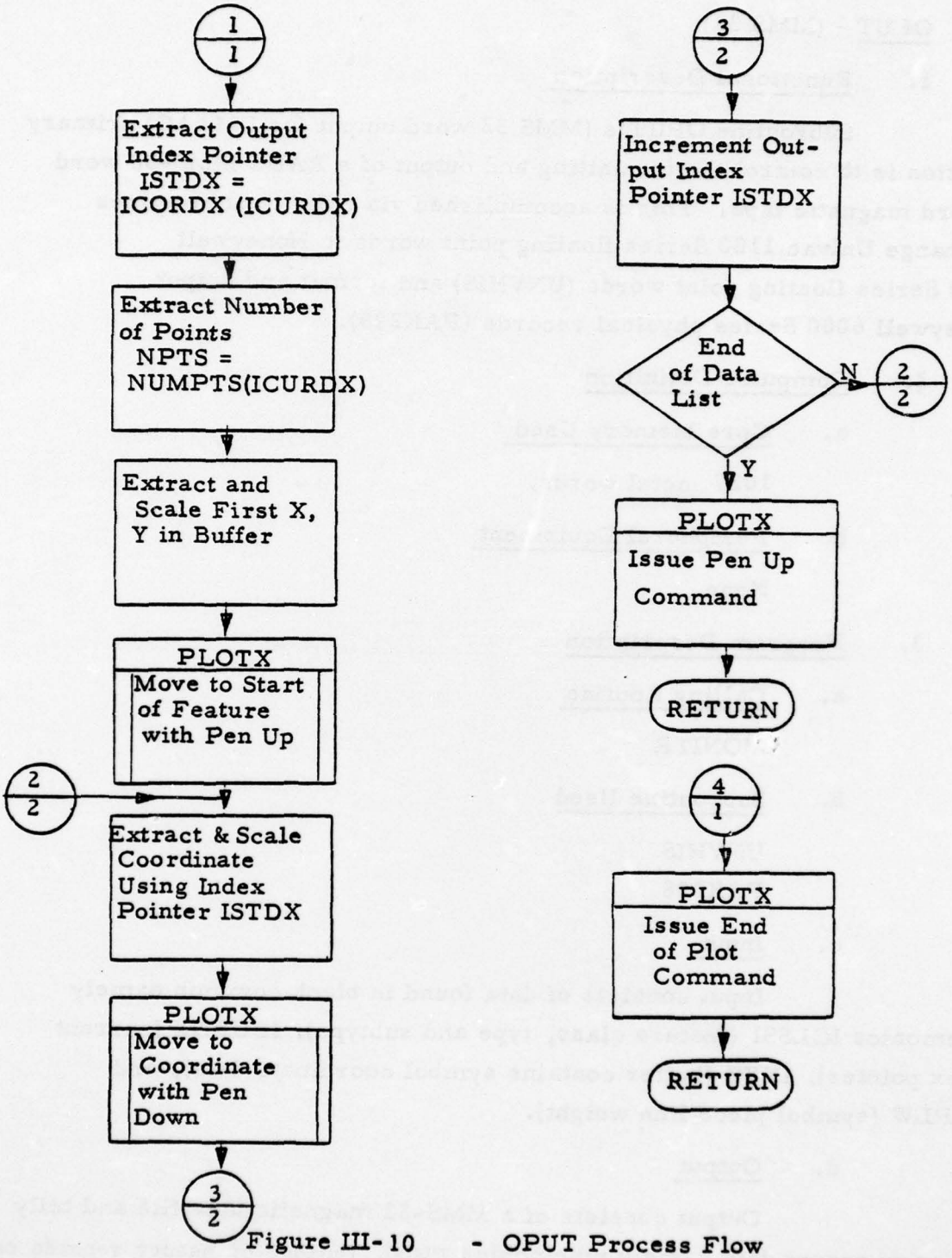



Figure III-10 - OPUT Process Flow

(Page 2 of 2)

J. OPUT - (MMS 32)

1. Functional Description

Subroutine OPUT's (MMS 32 word output for DMAAC) primary function is to control the formatting and output of a RADC MMS-32 word record magnetic tape. This is accomplished via calls to subroutines to change Univac 1100 Series floating point words to Honeywell 6000 Series floating point words (UNVHIS) and format and output Honeywell 6000 Series physical records (PAK298).

2. Computer Definition

a. Core Memory Used

1027 octal words.

b. Peripheral Equipment

None

3. Program Description

a. Calling Routine

MONITR

b. Subroutine Used

UNVHIS
PAK298

c. Input

Input consists of data found in blank common namely mnemonics ICLSS1 (feature class, type and subtype), ICURDX (current index pointer), IXYZ (buffer contains symbol coordinate data), and ISYPLW (symbol piece line weight).

d. Output

Output consists of a MMS-32 magnetic tape file and tally summary report data namely mnemonics IHEDOT (number of header records output),

NPTSOT (number of points output) and IDTOUT (number of data records output).

e. Processing Methodology

Processing flow of subroutine OPUT (MMS-32 word for DMAAC) is shown in Figure III- 11. Entry is made via GLSS control routine MONITR. Upon taking process control, subroutine OPUT interrogates a first entry flag (mnemonic M2). If found to be the first entry, a JCRS record (MMS-32 word start record) is formatted and packed into a Honeywell Series 6000 physical record via subroutine call to PAK298. Mnemonic M1 is then set to normal process for subsequent entries. The job through flag (IJBEND) is checked and if its status is "end of job" (IJBEND=2) a JCRE record (MMS-32 word end record) is formatted and output via subroutine call to PAK298. If the job end flag is not set to end of job, the minimum, maximum, first, last points are calculated for the symbol coordinate data in question. These points are then changed from Univac 1100 Series integer numbers to Univac 1100 Series floating point number and scaled to inches. Subroutine UNVHIS is called to change the Univac 1100 Series floating point numbers (min, max, first, last) to Honeywell 6000 Series floating point numbers. The Honeywell floating point numbers are then stored in the respective locations within the header record IHDREC. The symbol piece features class, type and subtype (ICLSS1) is entrant and stored into the header record along with the symbol piece line weight (ISYPLW). The header record is then packed into the physical record via subroutine PAK298. The symbol coordinate data are extracted from the input buffer IXYZ, changed to Univac floating point numbers and scaled to inches. The Univac floating point numbers are then altered to Honeywell floating point numbers via Subroutine UNVHIS. The Honeywell floating point numbers are stored

into a thirty two word array (IMMS MMS-32 word data records) and packed into the Honeywell physical record via PAK298. Upon reaching an end of input coordinate data indication, the partial data record, if one exists, is output via PAK298. Process control is then returned to the calling routine MONITR.

f. Calling Sequence

Call OPUT

g. Major Algorithms

None

4. Program Constants and Variables

ICLSS1 - feature class, type, subtype (Field Data)
IJBEND - job end flag (IJBEND=2)
IHDREC - MMS header record array (MMS 32 word)
IMMS - MMS 32 word data record
IHEDOT - number of header records output
NPTSOT - number of points output
IDTOUT - number of data records output

5. Error Conditions

None

**Subroutine OPUT**

MMS-32 word for DMAAC

Purpose- To format and output a HIS MMS-32 word record magnetic tape via subroutine UNVHIS and PAK298.

Input- Data found in blank common namely header data, symbol piece coord. data, etc.

Output- Magnetic tape containing nine MMS-32 word records per physical record (Subroutine PAK298)

Subroutine Used - UNVHIS-Univac floating point number to Honeywell 6000 floating point number.

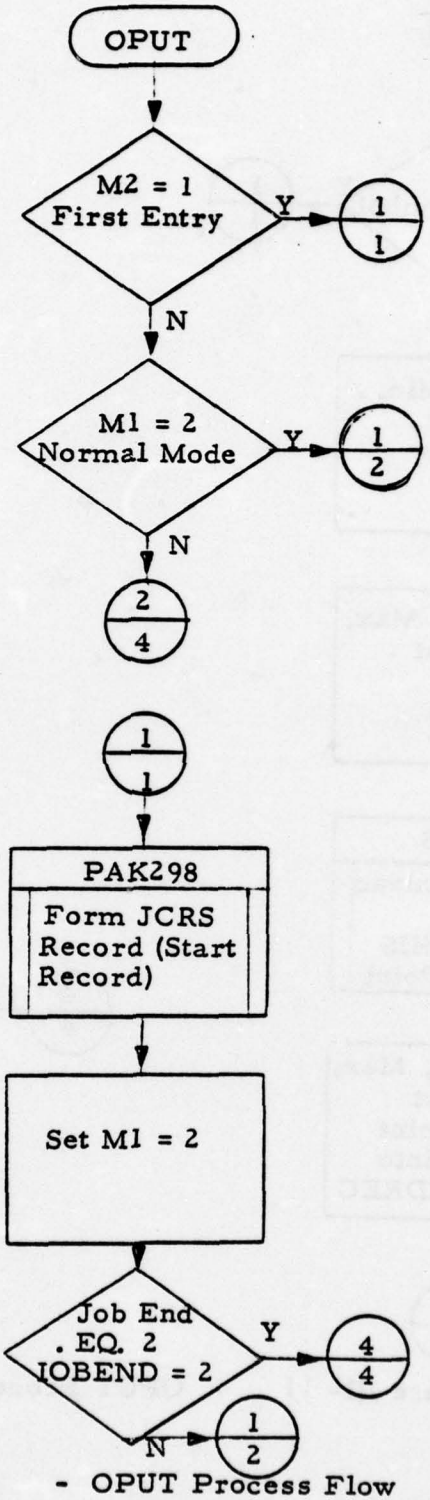



Figure III- 11 - OPUT Process Flow

(Page 1 of 4)

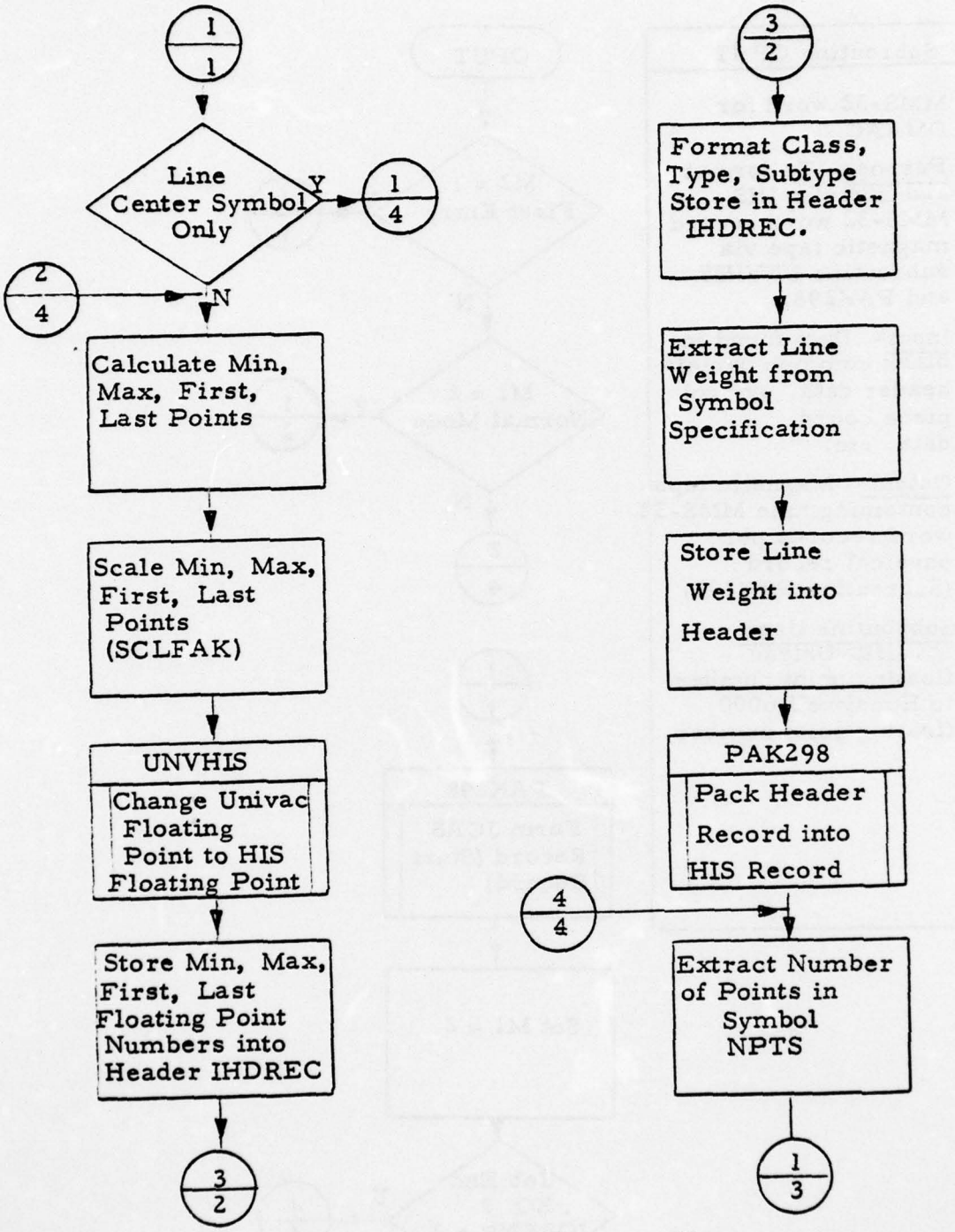



Figure III- 11 - OPUT Process Flow

(Page 2 of 4)

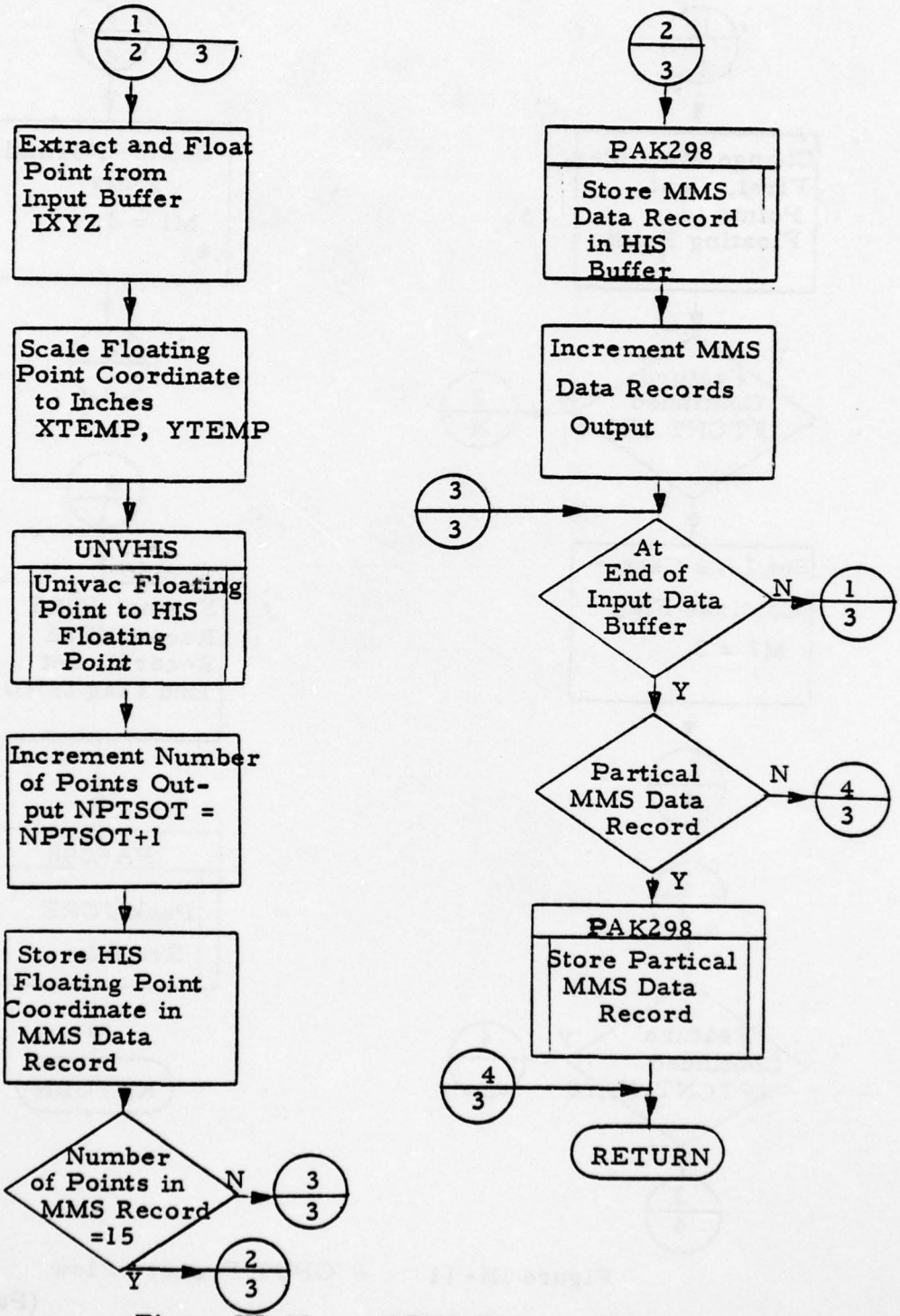



Figure III-11 - OPUT Process Flow

(Page 3 of 4)

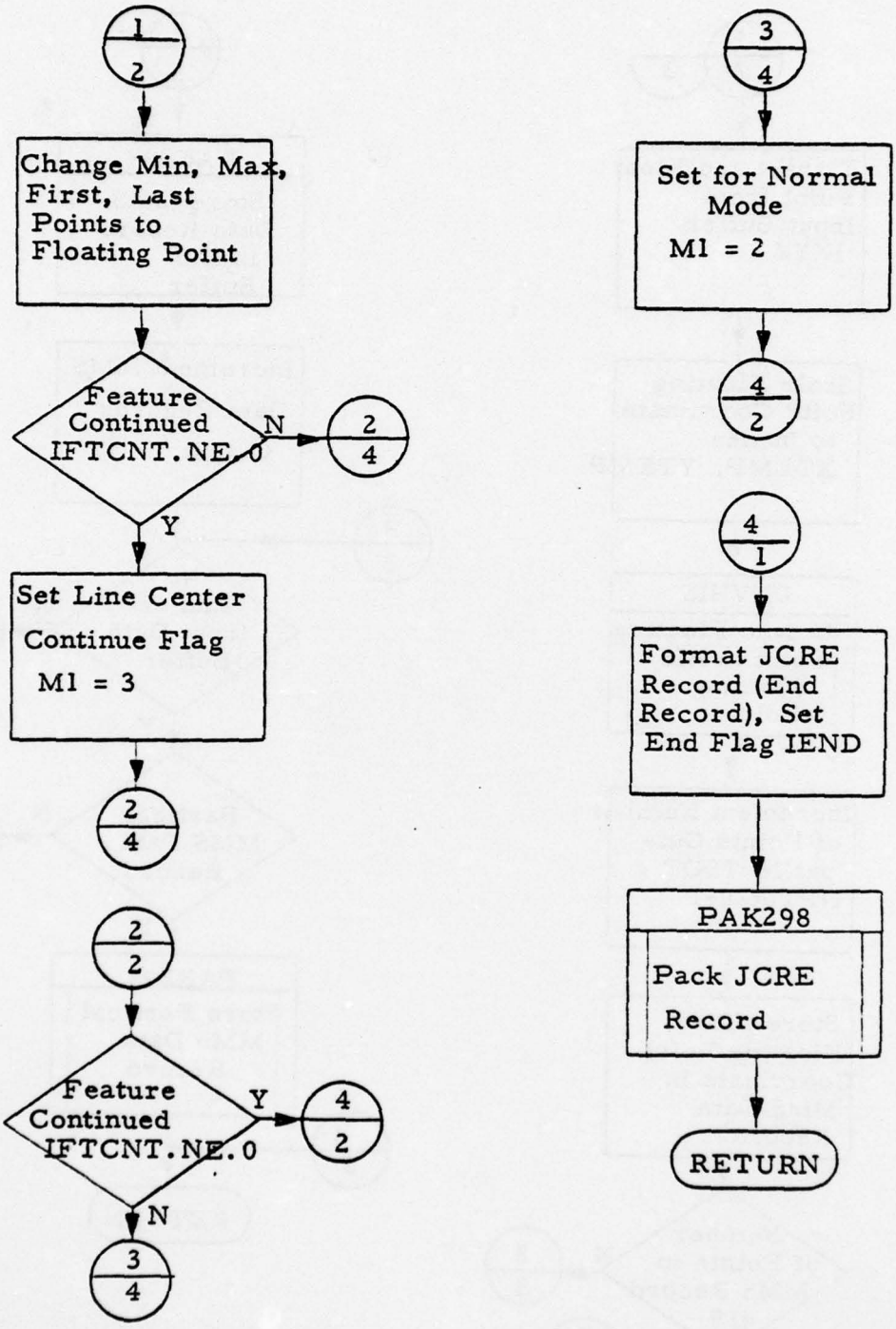



Figure III-11 - OPUT Process Flow

(Page 4 of 4)

## K. PAK298

### 1. Functional Description

The primary function of subroutine PAK298 is to format and pack MMS-32 word records into a 298 word buffer so its contents are identical to a Honeywell Series 6000/600 GCOS system standard formatted magnetic tape which are nine MMS-32 word record per 298 word physical record.

### 2. Computer Definition

#### a. Core Memory Used

705 octal words

#### b. Peripheral Equipment

Magnetic tape unit.

### 3. Program Description

#### a. Calling Routine

OPUT (MMS-32 word output routine for DMAAC)

#### b. Subroutine Used

Univac Fortran I/O processing routine NTRAN

#### c. Input

Input consists of a MMS-32 word header records and data record via calling sequence. Also input via calling sequence is a flag indication end of process.

#### d. Output

Output consists of a magnetic tape formatted identical to a Honeywell Series 6000/600 GCOS system standard tape containing nine MMS-32 word records per 298 word physical record. See Figure III-12 for the magnetic tape format. See Figure III-13 for format of a 298 word physical record.

e. Processing Methodology

Subroutine PAK298 processing flow is depicted in Figure III- 14. Entry is made via Subroutine OPUT (MMS-32 word output routine for DMAAC). Upon first entry, a Honeywell Series 6000/600 GCOS System fourteen word header label record (containing zeroes) is written to magnetic tape followed by a standard end of file mark. A block control word containing the block serial number and block size (451 octal) is formatted and stored into the first word of the physical record (see Figure III- 13). The data input via the calling sequence (mnemonic IMMS) is extracted and stored into the physical record (mnemonic IREC). The record control word (mnemonic IRCW) is stored into buffer IREC via record control index pointer IRCWDX. If IREC becomes full (mnemonic INDEX equals 298) or end of job indicator is set (mnemonic IEND ) the 298 physical record is written to magnetic tape via UNIVAC FORTRAN I/O processing routine NTRAN. If the end of job indicator is set, a standard end of file is written via NTRAN and the magnetic tape rewound. If the above conditions are false, the record control index pointer (IRCWDX) is incremented by thirty three and buffer IREC index pointer is incremented by one with process control being returned to the calling routine OPUT.

f. Calling Sequence

Call PAK298 (IMMS, IEND)

IMMS - MMS-32 word record (header followed by data records).

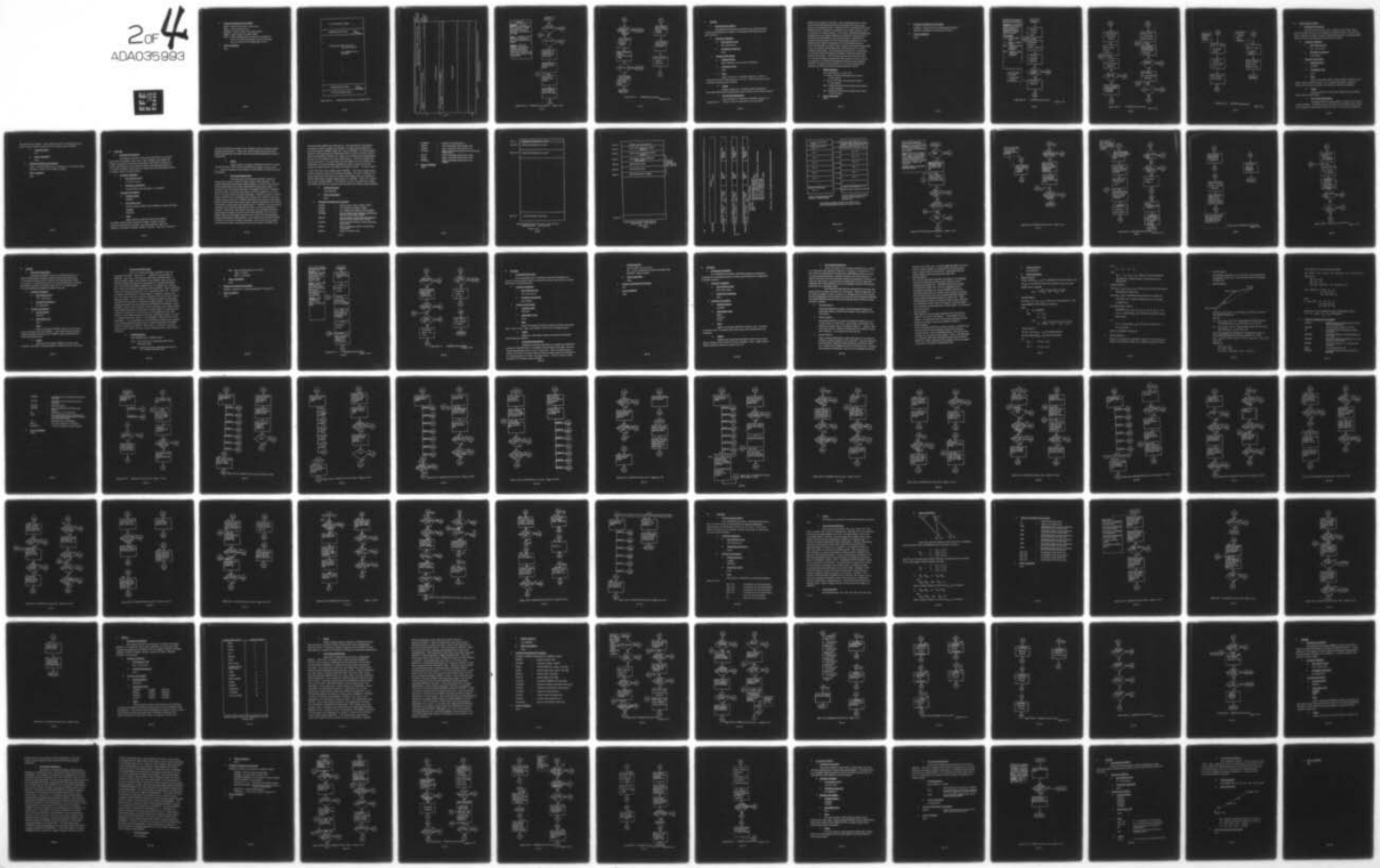
IEND - Flag to indicate end of process
IEND = 0 normal
IEND = 1 end of process.

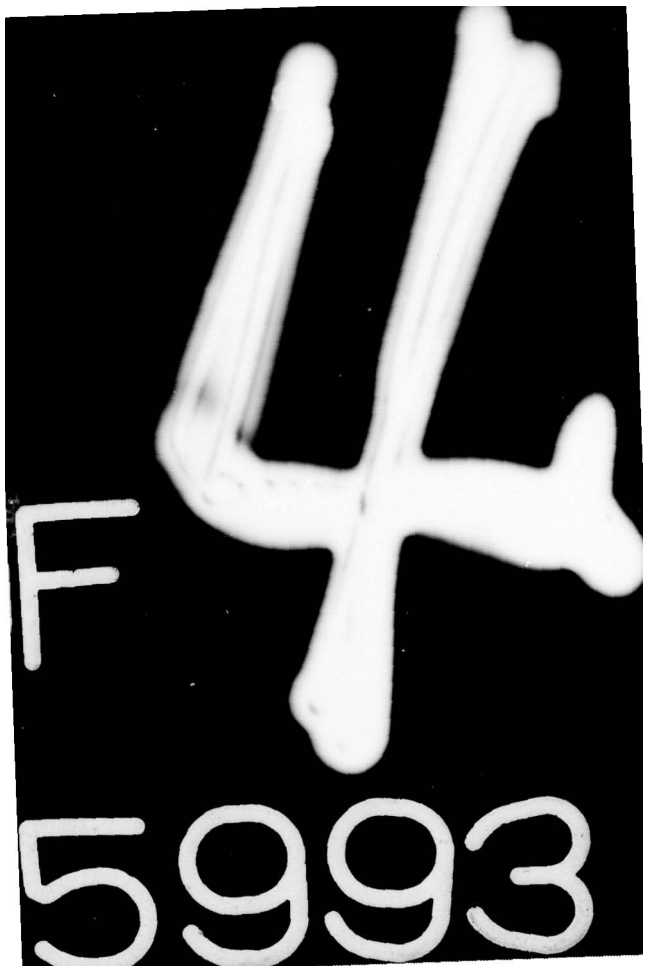
g. Major Algorithms

None

AD-A035 993 PRC INFORMATION SCIENCES CO MCLEAN VA F/G 8/2
ACS SYMBOLIZATION FOR DMAAC. VOLUME II. COMPUTER PROGRAM DOCUME--ETC(U)
NOV 76 P D BELL, J A NEUFFER, M L TAYLOR F30602-75-C-0319
UNCLASSIFIED RADC-TR-76-334-VOL-2 NL

2 OF 4
ADA035993




F
4
5993

4. Program Constants and Variables

IBCW - physical data block control word.

IRCW - record control word.

IRCWDX - record control word index pointer.

INDEX - index pointer into buffer IREC.

IREC - buffer containing block control word followed by nine thirty three word logical records containing record control word and MMS input record.

5. Error Conditions

None

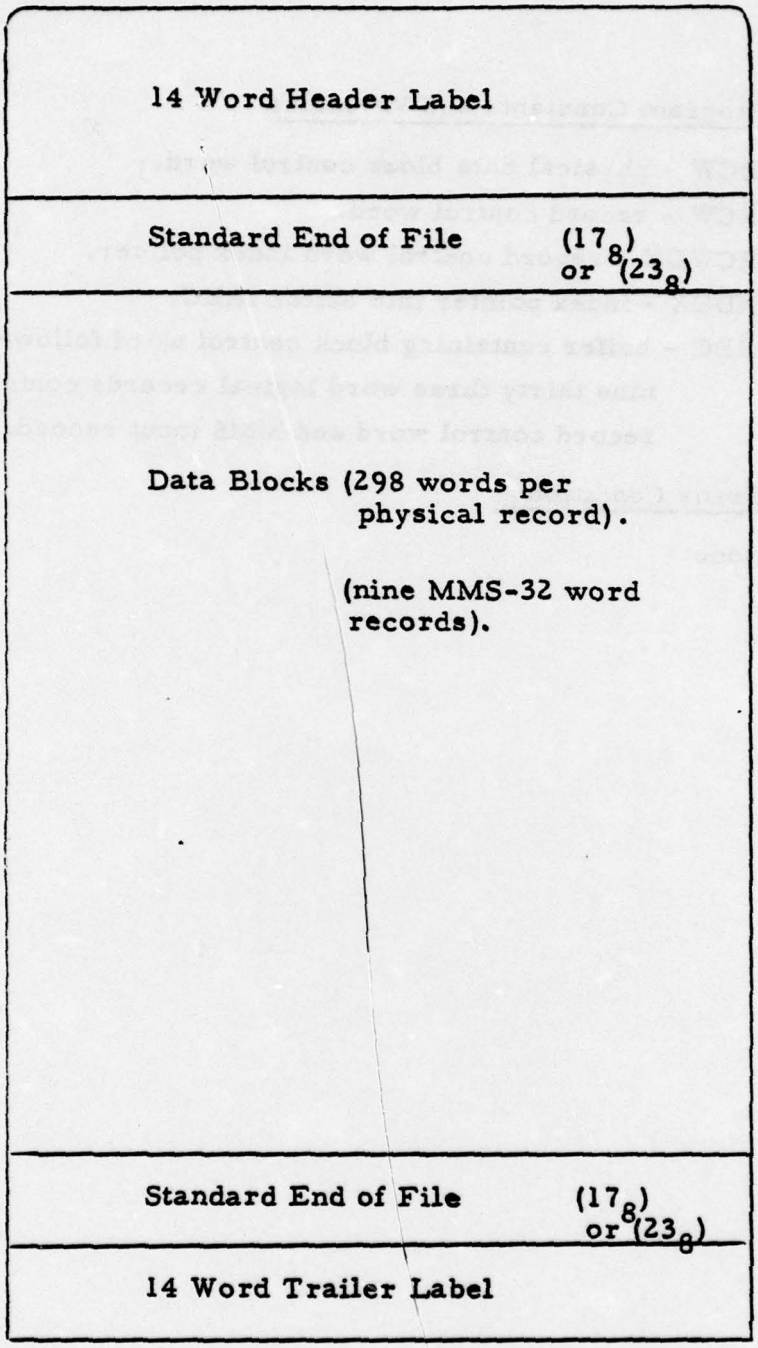



Figure III- 12 - HIS Standard Format of Labeled Files

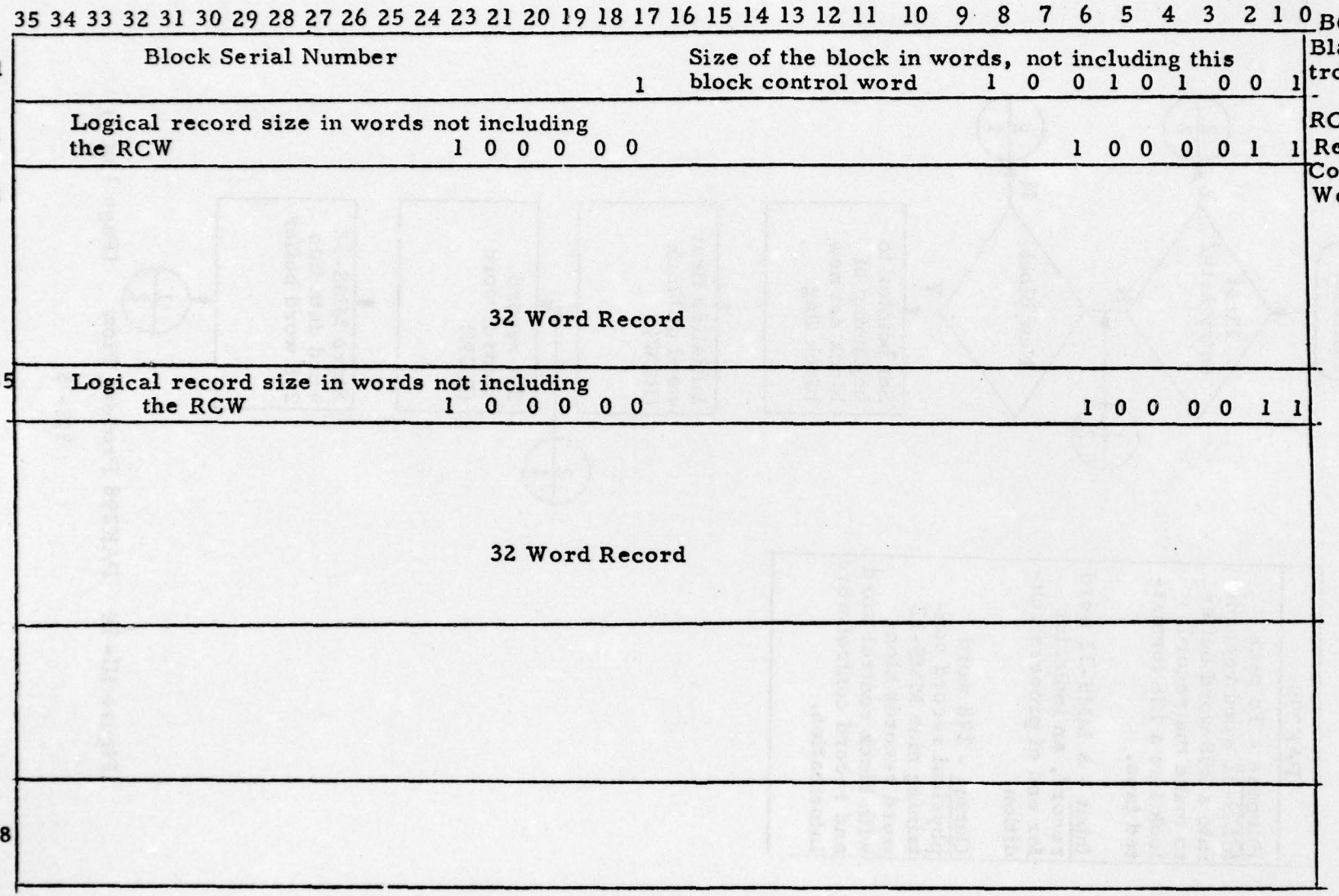



Figure III- 13 - Two Hundred and Ninety Eight Words Per Physical Record
Nine 32-Word Records

**PAK298**

**Purpose** - To pack MMS-32 word records into a 298 word buffer to make the record look like a HIS formatted tape.

**Input** - A MMS-32 word record, an indicator for end of process condition.

**Output** - 298 word physical record containing nine MMS-32 word records along with block control word and record control word information.

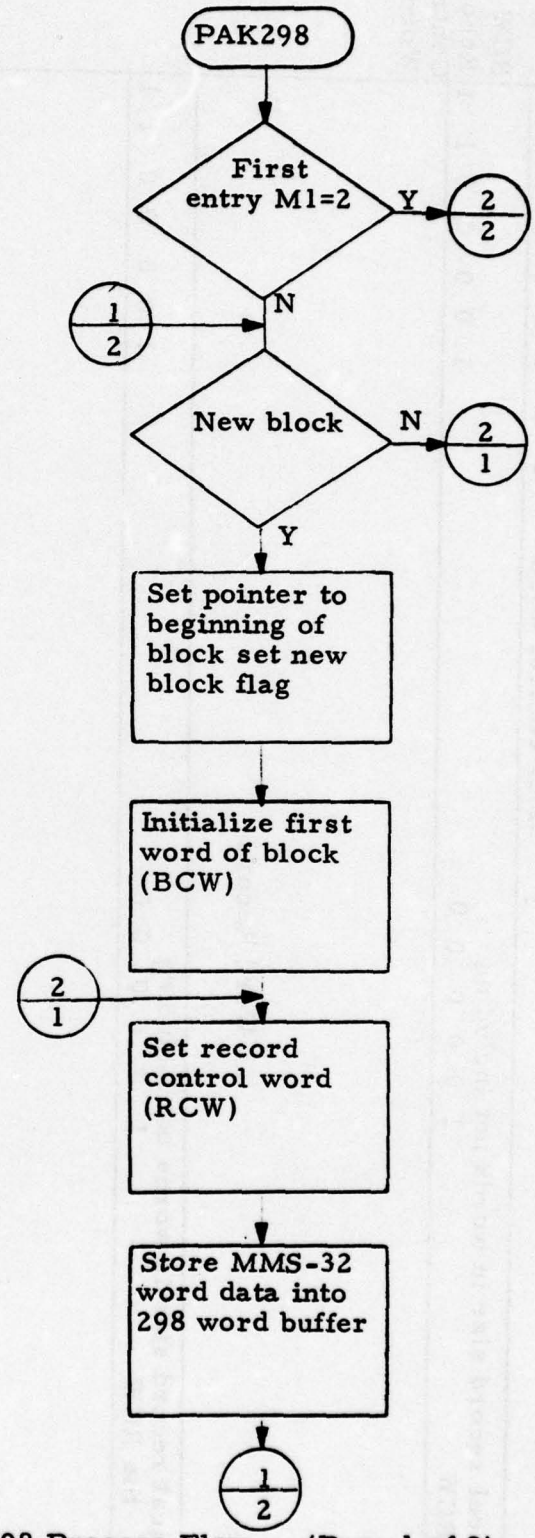



Figure III-14 PAK298 Process Flow (Page 1 of 2)

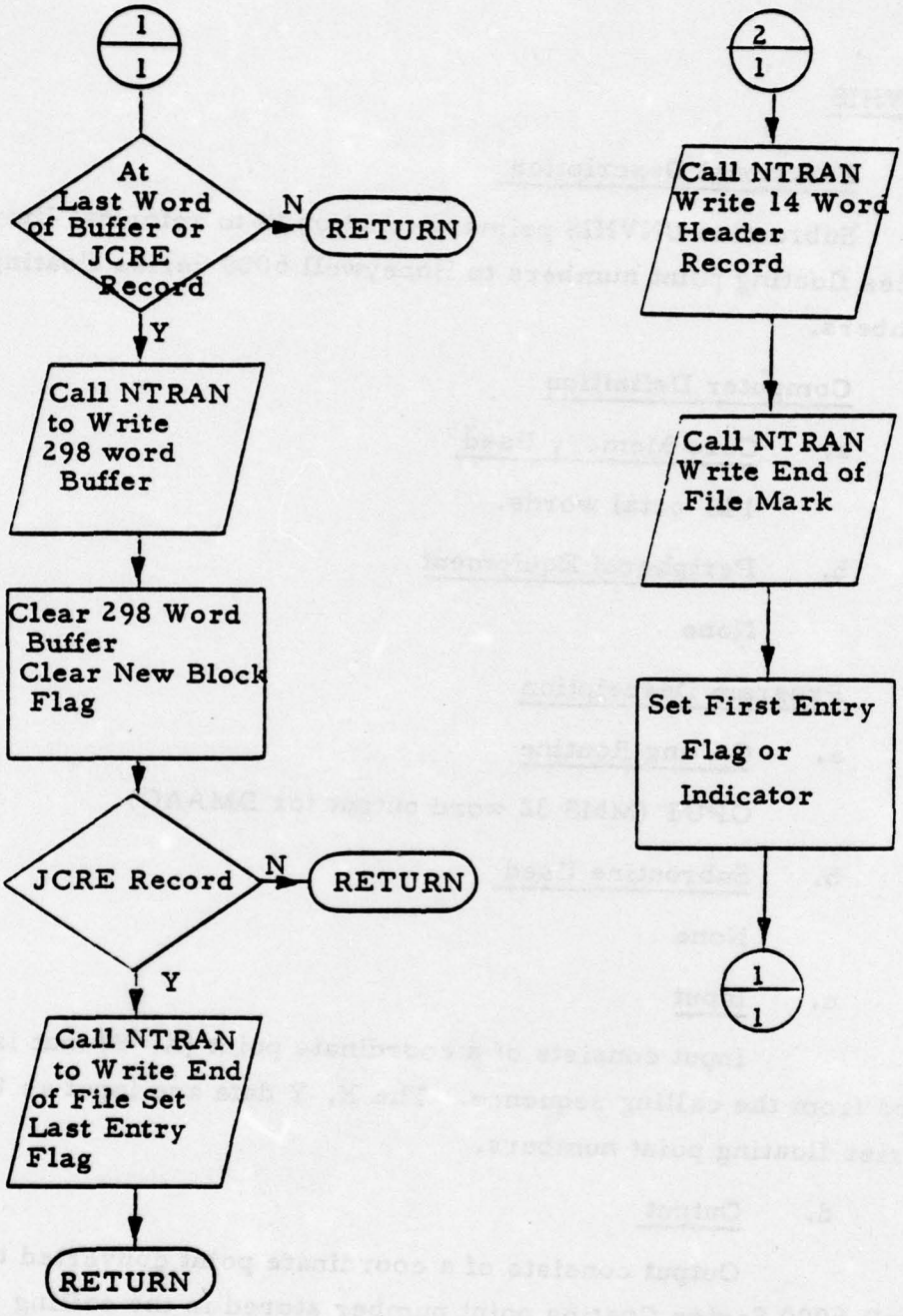



Figure III- 14 - PAK298 Process Flow

(Page 2 of 2)

## L. UNVHIS

### 1. Functional Description

Subroutine UNVHIS primary function is to reformat Univac 1100 Series floating point numbers to Honeywell 6000 Series floating point numbers.

### 2. Computer Definition

#### a. Core Memory Used

140 octal words.

#### b. Peripheral Equipment

None

### 3. Program Description

#### a. Calling Routine

OPUT (MMS 32 word output for DMAAC)

#### b. Subroutine Used

None

#### c. Input

Input consists of a coordinate point (X, Y) that is extracted from the calling sequence. The X, Y data are input as Univac 1100 Series floating point numbers.

#### d. Output

Output consists of a coordinate point converted to Honeywell 6000 Series floating point number stored in the calling sequence.

#### e. Processing Methodology

Processing flow of Subroutine UNVHIS is depicted in Figure III-15. Entry is made via subroutine call from OPUT

(MMS 32 word output for DMAAC). Upon receiving process control, subroutine UNVHIS extracts the X value floating point number and stores it into Mnemonic VAL. Mnemonic VAL is equivalenced (Fortran EQUIVALENCE statement) to mnemonic IVAL, thus treating in value in question as an integer number. The Univac 1100 Series floating point characteristic and mantissa (ICHART and IMANTS respectively) are extracted and reformatted. If the characteristic is less than one hundred and twenty eight (128) the characteristic (ICHART) is reformatted and the negative characteristic bit (Honeywell) turned on. If the characteristic is greater than or equal to one hundred twenty eight (128), one hundred twenty eight (128) is subtracted from it and the resultant reformatted. The reformatted characteristic and mantissa are OR-ed (Fortran OR Statement) together forming the Honeywell floating point number. If the X value was the last value processed, it is set into the output argument list (IOX) and the Y value is moved to mnemonic VAL with the aforementioned process being repeated. After the Y value has been processed, it is stored into the output argument list (IOY) and process control returned to OPUT.

f. Calling Sequence

CALL UNVHIS (X, Y, IOX, IOY)

X - Univac 1100 Series floating point number,
X value (Input)

Y - Univac 1100 Series floating point number,
Y value (Input)

IOX - Honeywell 6000 Series floating point number
X value (Output)

IOY - Honeywell 6000 Series floating point number
Y value (Output)

g. Major Algorithms

None

4. **Program Constants and Variables**

ICHART - Characteristic of floating point number
IMANTS - Mantissa of floating point number

5. **Error Conditions**

None

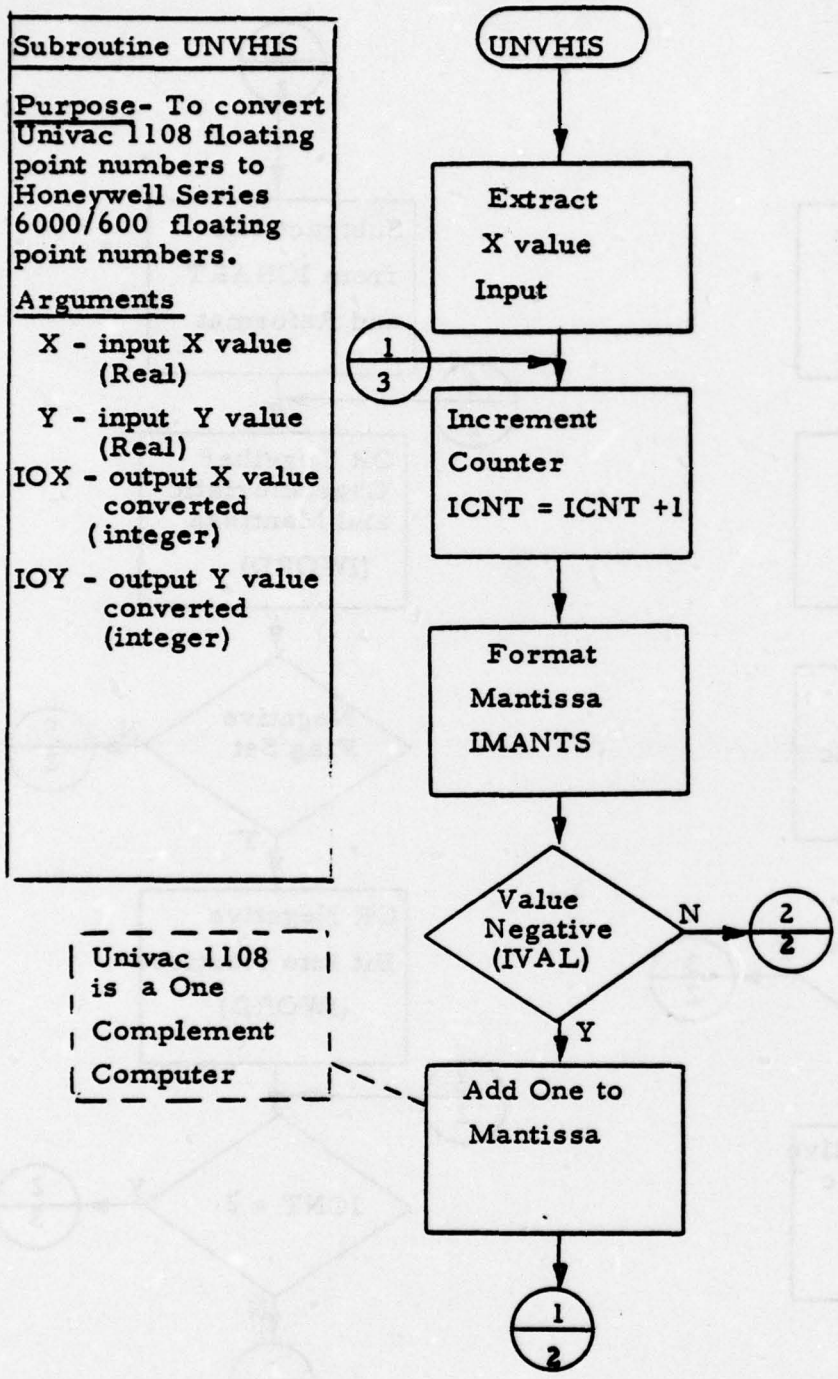



Figure III- 15 - UNVHIS Process Flow

Page 1 of 3

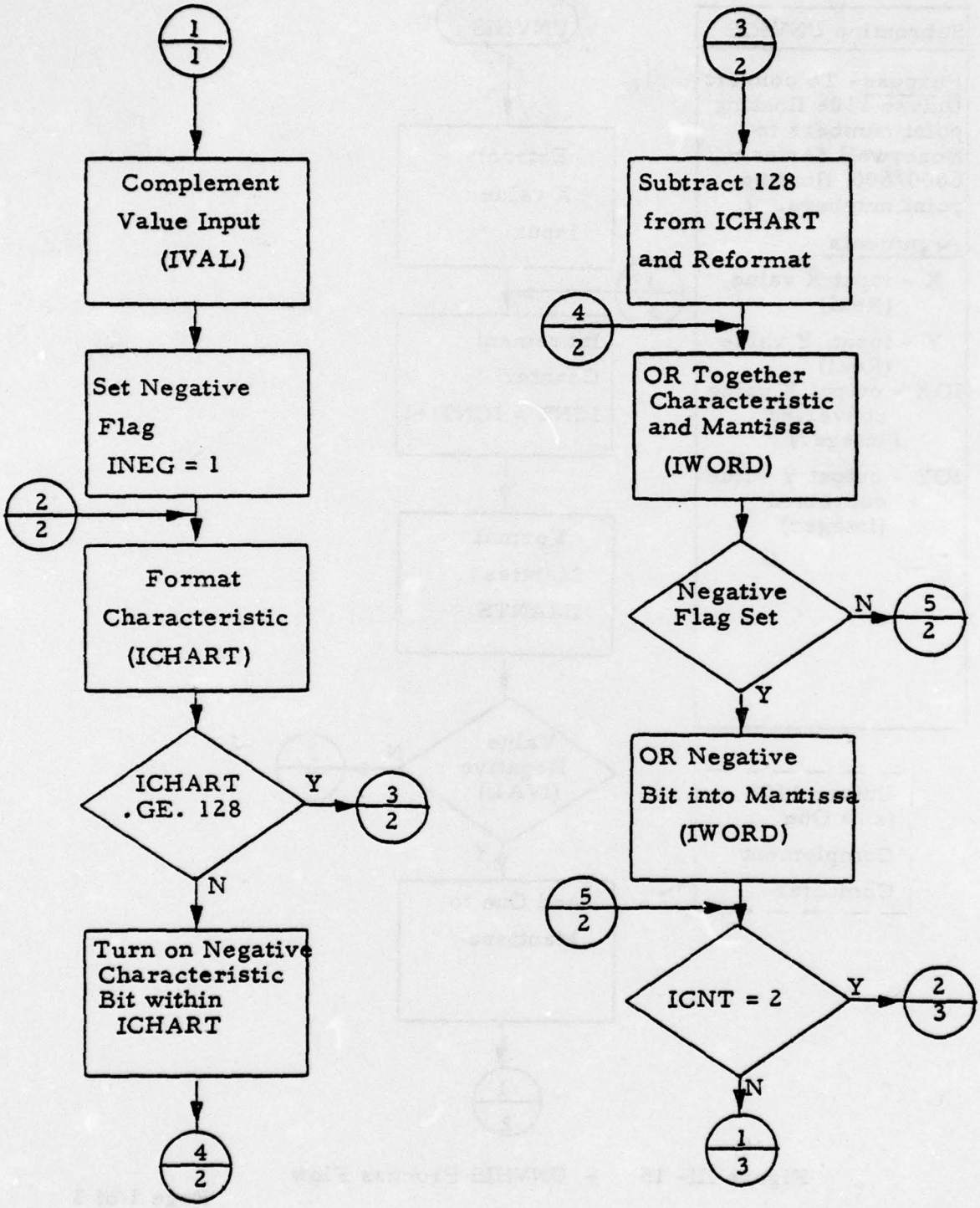



Figure III- 15 - UNVHIS Process Flow

Page 2 of 3

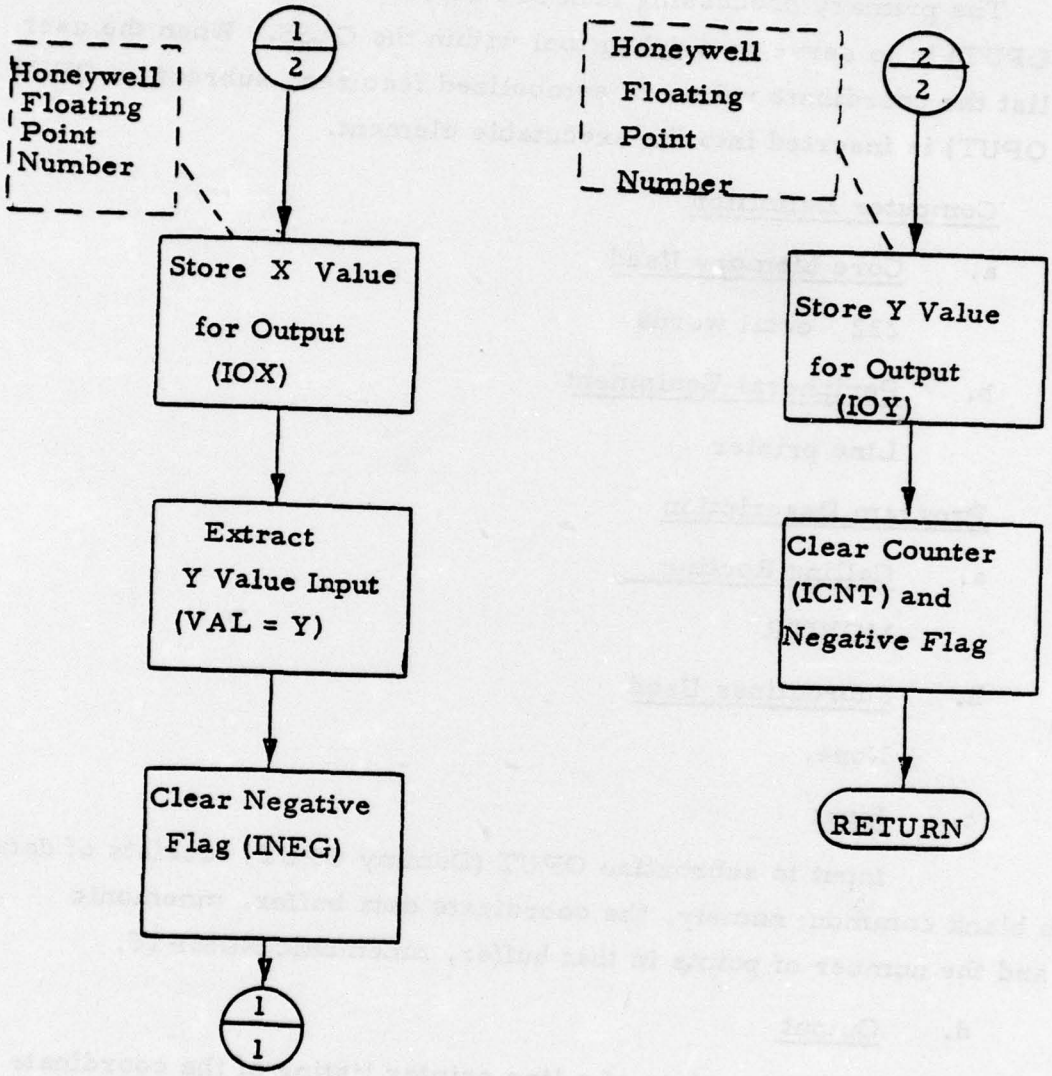



Figure III- 15 - UNVHIS Process Flow

## M. OPUT (Dummy OPUT)

### 1. Functional Description

The primary processing task of FORTRAN subroutine OPUT (Dummy OPUT) is to serve as a debug tool within the GLSS. When the user needs to list the coordinate values of symbolized features, subroutine OPUT (Dummy OPUT) is inserted into the executable element.

### 2. Computer Definition

#### a. Core Memory Used

232 octal words

#### b. Peripheral Equipment

Line printer

### 3. Program Description

#### a. Calling Routine

MONITR

#### b. Subroutines Used

None.

#### c. Input

Input to subroutine OPUT (Dummy OPUT) consists of data found in blank common; namely, the coordinate data buffer, mnemonic IXYZ, and the number of points in that buffer, mnemonic NUMPTS.

#### d. Output

Output consists of a line printer listing of the coordinate values found in buffer IXYZ.

#### e. Processing Methodology

Subroutine OPUT (Dummy OPUT) is called via the control routine MONITR. Coordinate data is extracted from the IXYZ buffer stored into a temporary output buffer mnemonic IOUT, in sets of six coordinates,

and output the line printer. Upon reaching an end of coordinate data indication, process control is returned to the calling routine MONITR.

f. Calling Sequence

N/A

g. Major Algorithms

None

4. Program Constants and Variables

IXYZ - Two dimensional array containing X, Y coordinate points.
NUMPTS - Number of data points to output.

5. Error Conditions

None.

## N. FETCOR

### 1. Functional Description

The primary function of subroutine FETCOR is to correlate input feature descriptor codes (class, type, subtype, and codified descriptors) against stored feature descriptor codes with a resultant of a random index pointer used to retrieve symbol specification directives. A secondary function of FETCOR is to report non-correlated input feature descriptor codes to the line printer.

### 2. Computer Definition

#### a. Core Memory Used

2114 octal words.

#### b. Peripheral Equipment

Two disk files (one sequential, one random).

### 3. Program Description

#### a. Calling Routine

MONITR

#### b. Subroutines Used

System routine DEFINE FILE (initialize random disk file)

CKDESP
NOCORR

#### c. Input

Input consists of blank common areas (feature descriptor data) namely mnemonics ICLSS1, ICLSS2, ICLSS3, (symbol specification overrides), and (status indicator flags and pointers), mnemonics IOTHED, IDIRCT, IOVRID. Input also consists of

symbol specification descriptor codes (DESP) and their respective symbol specification directives. Tables III-1 and III-2 depict the format of the symbol specification descriptor codes and symbol specification directives, respectively.

d. Output

Output consists of symbol specification directives stored in blank common areas (symbol specification directives common area) or (feature descriptor data) mnemonic buffer IHEAD (currently not used at DMAAC)

e. Processing Methodology

Processing flow of subroutine FETCOR is shown in Figure III-16. Entry is made via GLSS control routine MONITR. The symbol specification directives common buffers are cleared with the read symbol specification from header flag (IDIRCT) being interrogated. If found to be set, the symbol directives are unpacked from their respective locations found in the header text buffer (not used at DMAAC) (see Table III-3), and stored in the following mnemonics: NUMPEC (number of symbol pieces), ISTNO1 (color separation sheet number one), ISTNO2 (color separation sheet number two), ICONON (conformal/non-conformal information buffer), ISYTP (symbol piece type buffer), ISYSZ (symbol piece size buffer), and ISYPLW (symbol piece line weight buffer). If IDIRCT is not set (normal at DMAAC) the symbol specification directive override flag (IOVRID) is checked. If found set, the override descriptors are examined against the input feature descriptor via call to subroutine CKDESP, and if a match exists, (IHIT=0) the matched overrides are stored in the aforementioned symbol specification common buffer. If the symbol override flag (IOVRID) is not set or the above override check fails, a first entry check is made. If it is the first entry, the previously stored descriptors (via SPEC) are

read into buffer IDESP (see Table III-3). The input feature descriptors, mnemonics ICLSS1, ICLSS2, and ICLSS3, are then checked against the above descriptors (IDESP), again via a call to CKDESP. If a match exists, (IHIT=0), the index generated (mnemonic IDX) is used to retrieve from the symbol specification random file a unique set of symbol piece specification directives to be applied to the given input feature. Table III-4 depicts the above correlation. If a match does not exist (IHIT=1), no correlation flag (IERRID) is set and subroutine NOCORR is called to report on the line printer, the input feature's descriptor and feature number. Process control is then returned to the calling routine MONITR. If the store symbol specification in header flag is set (mnemonic IOTHED, not used at DMAAC), the above symbol piece directives are packed into header text area mnemonic IHEAD with process control being returned to MONITR. Table III-3 depicts the format of buffer IHEAD. If IOTHED is not set, the symbol piece types are tallied with process control being returned to the calling routine MONITR.

f. Calling Sequence

Call FETCOR

g. Major Algorithms

None

4. Program Constants and Variables

| | | |
|---|---|---|
| ICLSS1 | - | feature class, type, subtype (BCD) |
| ICLSS2 | - | six codified descriptors (BCD) |
| ICLSS3 | - | two codified descriptors (left justified) |
| IOTHED | - | flag to indicate pack symbol specification directive into buffer IHEAD. |
| IDIRCT | - | flag to indicate unpack symbol specification directives from buffer IHEAD. |
| IOVRID | - | flag to indicate feature symbol specification exists. |
| IDESP | - | buffer containing symbol specification descriptors. |
| IHEAD | - | buffer for header text. |

| | | |
|---|---|---|
| NUMPEC | - | number of symbol pieces. |
| ISTNO1 | - | color separation sheet number one. |
| ISTNO2 | - | color separation sheet number two. |
| ICONON | - | buffer containing conformal/non-conformal information. |
| ISYTP | - | buffer containing symbol piece type. |
| ISYSZ | - | buffer containing symbol piece size. |
| ISYPLW | - | buffer containing symbol piece line weights. |

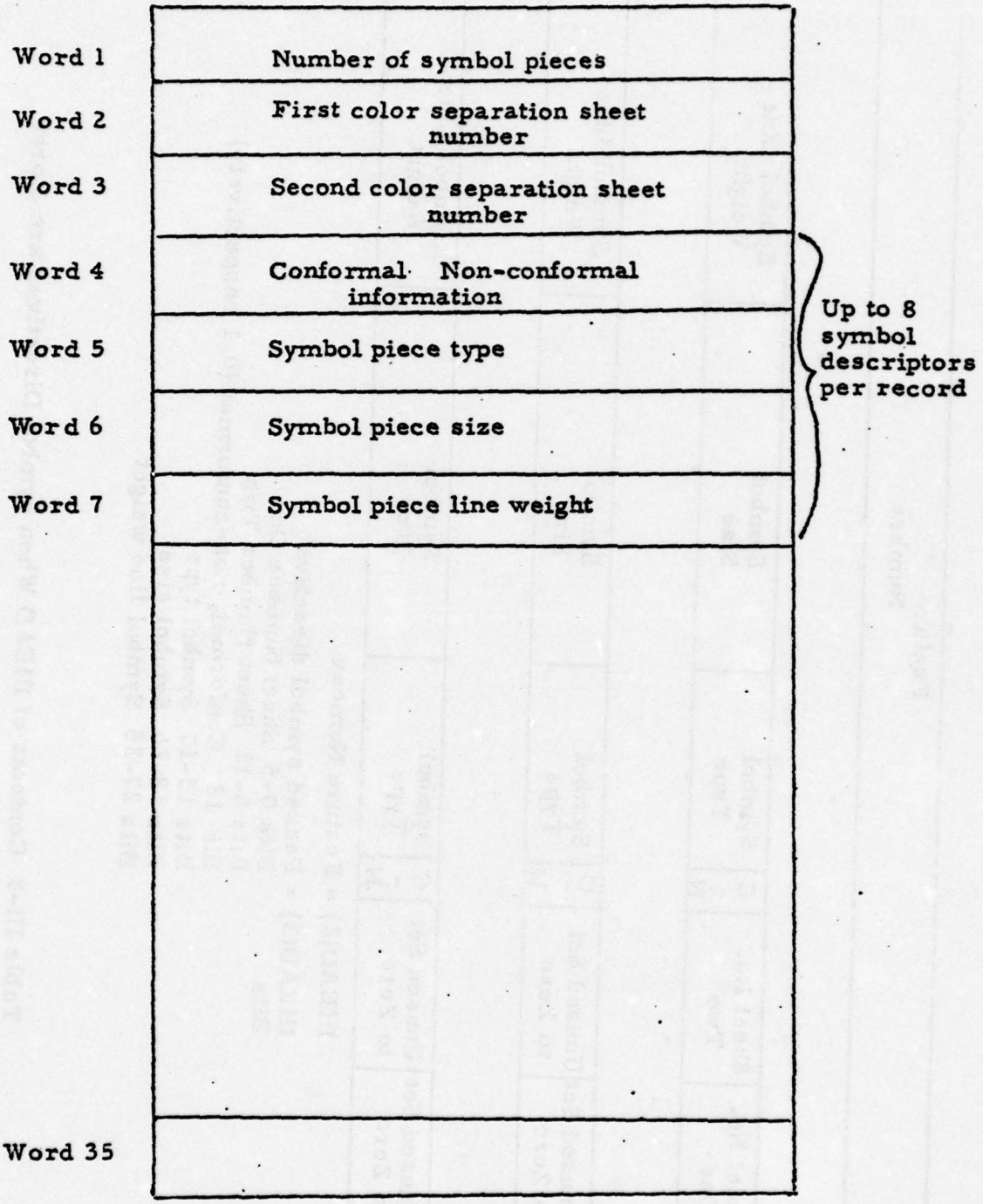
5. Error Conditions

None

| | |
|---|---|
| Word 1 | Number of descriptor in file |
| Word 2-4 | First descriptor code input |
| Word 5-7 | Second descriptor code input |
| | . |
| | . |
| | . |
| | . |
| | . |
| | . |
| | . |
| | . |
| | . |
| | . |
| | . |
| | . |
| (N-3)-N | Last descriptor code input |

Symbol Specification Descriptor Code Format
Sequential File File Code 08

Table III-1




Symbol Specification Random Record
Random File File Code 02

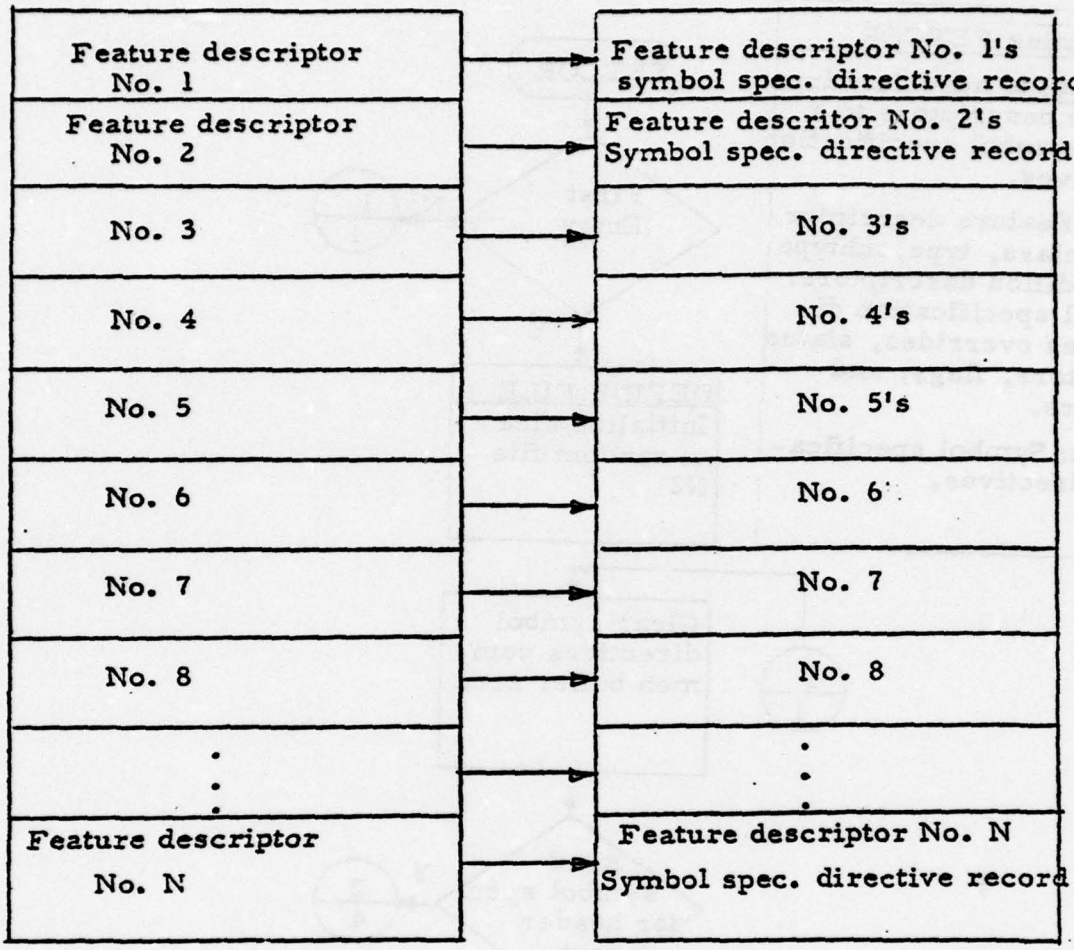
Table III-2

| Bit | 0 1 2 3 4 5 6 7 8 9 10 11 | 12 | 13 14 15 16 17 | 18 19 20 21 22 23 24 25 26 | 27 28 29 30 31 32 33 34 35 |
|---|---|---|---|---|---|
| IHEAD(2) | Feature Number | | | | |
| IHEAD(3) | Sheet No. One | Sheet No. Two | C - N | Symbol Type | Symbol Size | Symbol Line Weight |
| IHEAD(4) | Unused Set to Zero | Unused Set to Zero | C - N | Symbol Type | Symbol Size | Symbol Line Weight |
| IHEAD(10) | Unused Set to Zero | Unused Set to Zero | C - N | Symbol Type | Symbol Size | Symbol Line Weight |

IHEAD(2) = Feature Number

IHEAD(3) = Packed symbol directive
Bits
Bits 0-5 Sheet Number One
Bits 6-11 Sheet Number Two
Bit 12 Conformal, non-conformat (0, 1 respectively)
Bits 13-17 Symbol type
Bits 18-26 Symbol size
Bits 27-35 Symbol line weight

Table III-3 Contents of IHEAD When Symbol Directives are Stored




Feature descriptors as they appear in buffer IDESP

Symbol specification directive records as they appear on random file.

Correlation between feature descriptors and symbol specification directive random records.

Table III-4

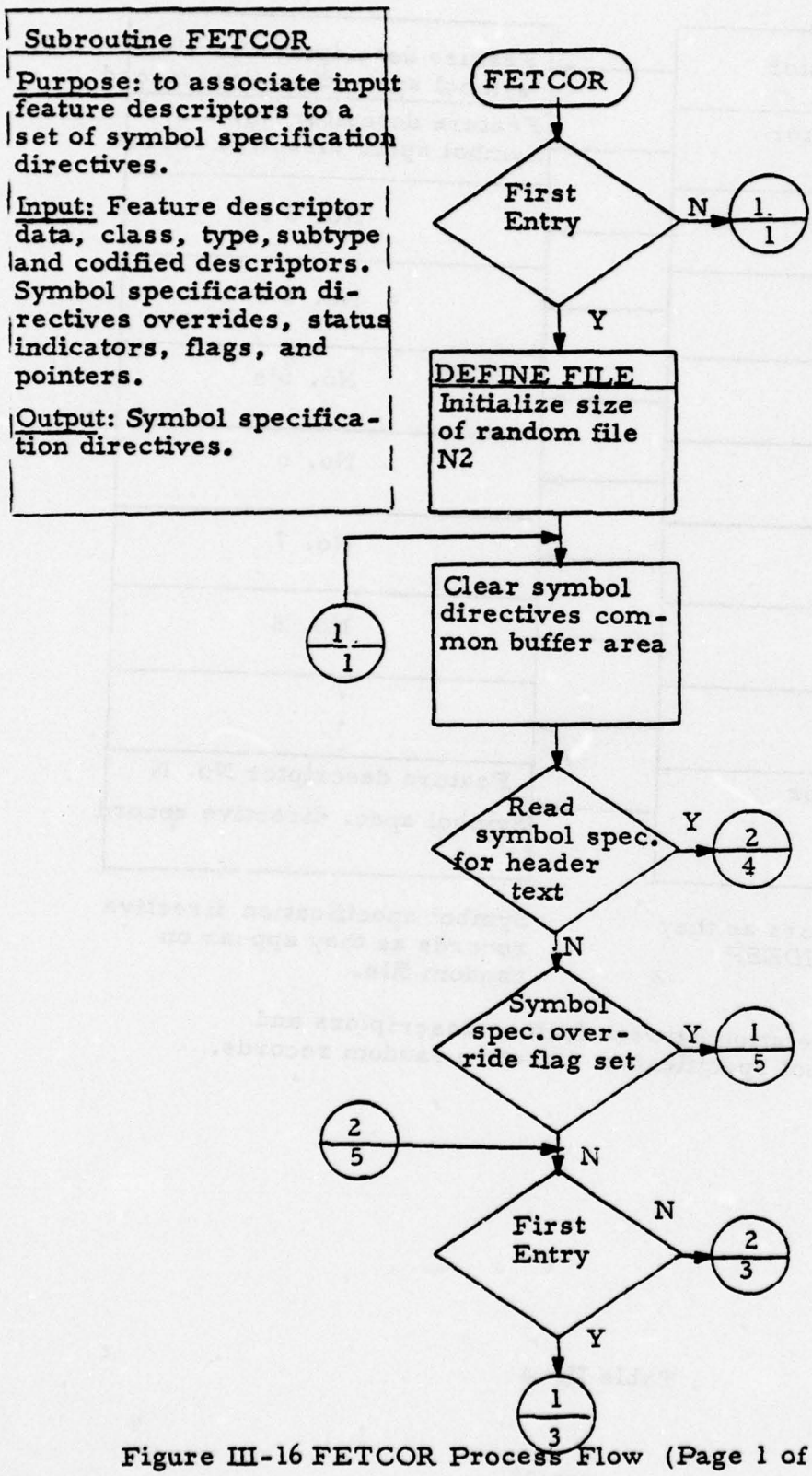



Figure III-16 FETCOR Process Flow (Page 1 of 5)

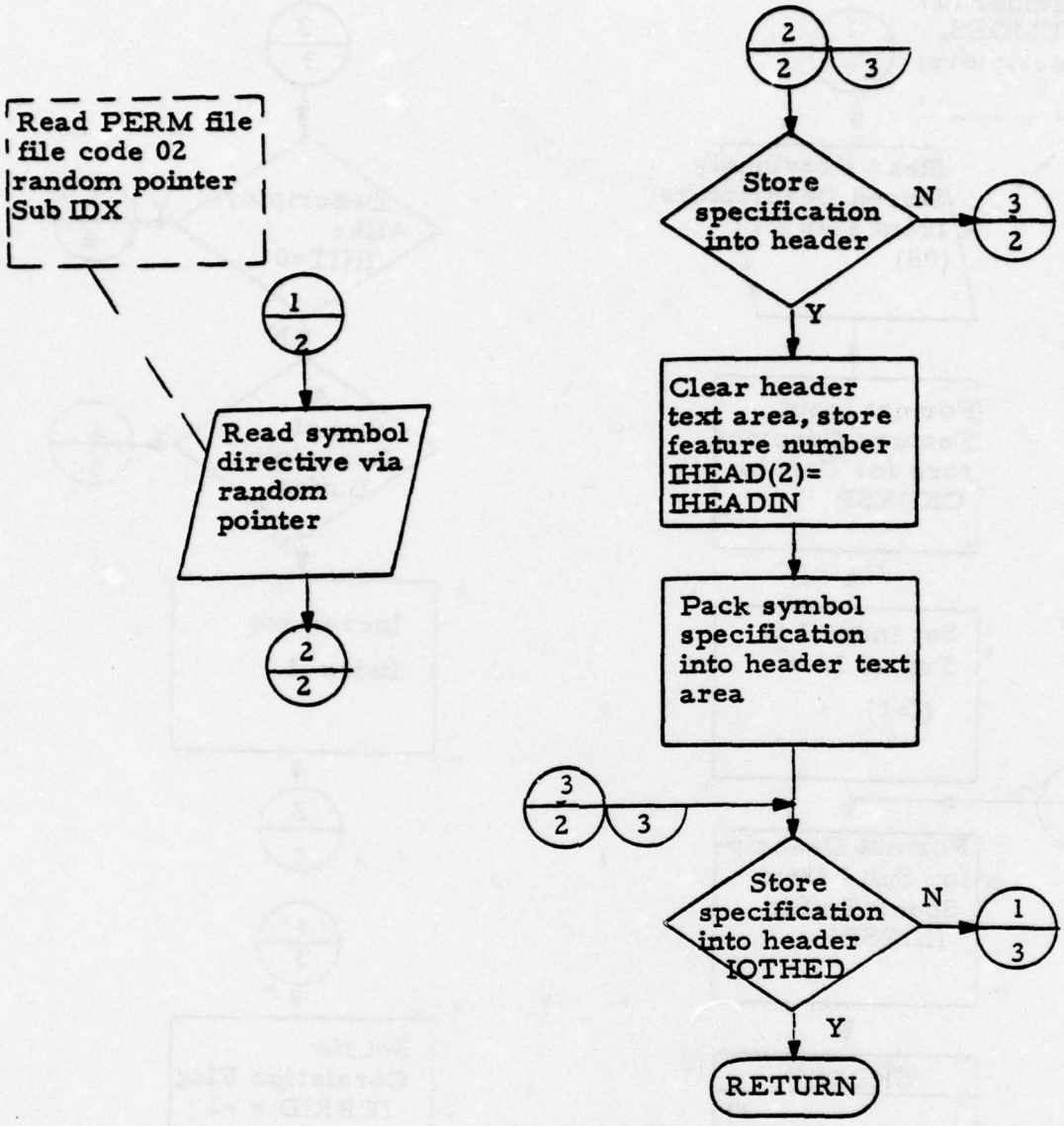



Figure III-16 FETCOR Process Flow (Page 2 of 5)

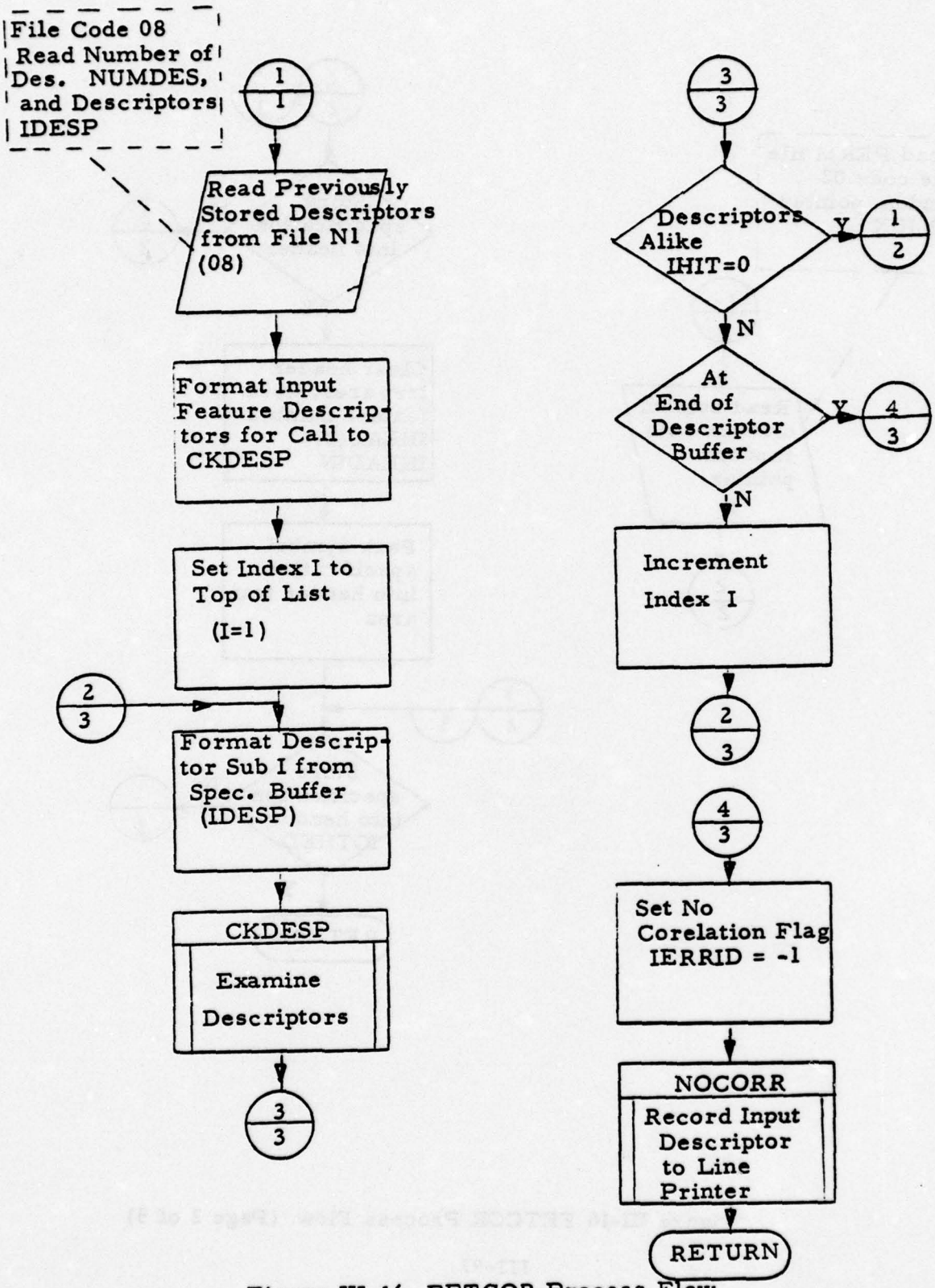



Figure III-16 FETCOR Process Flow (Page 3 of 5)

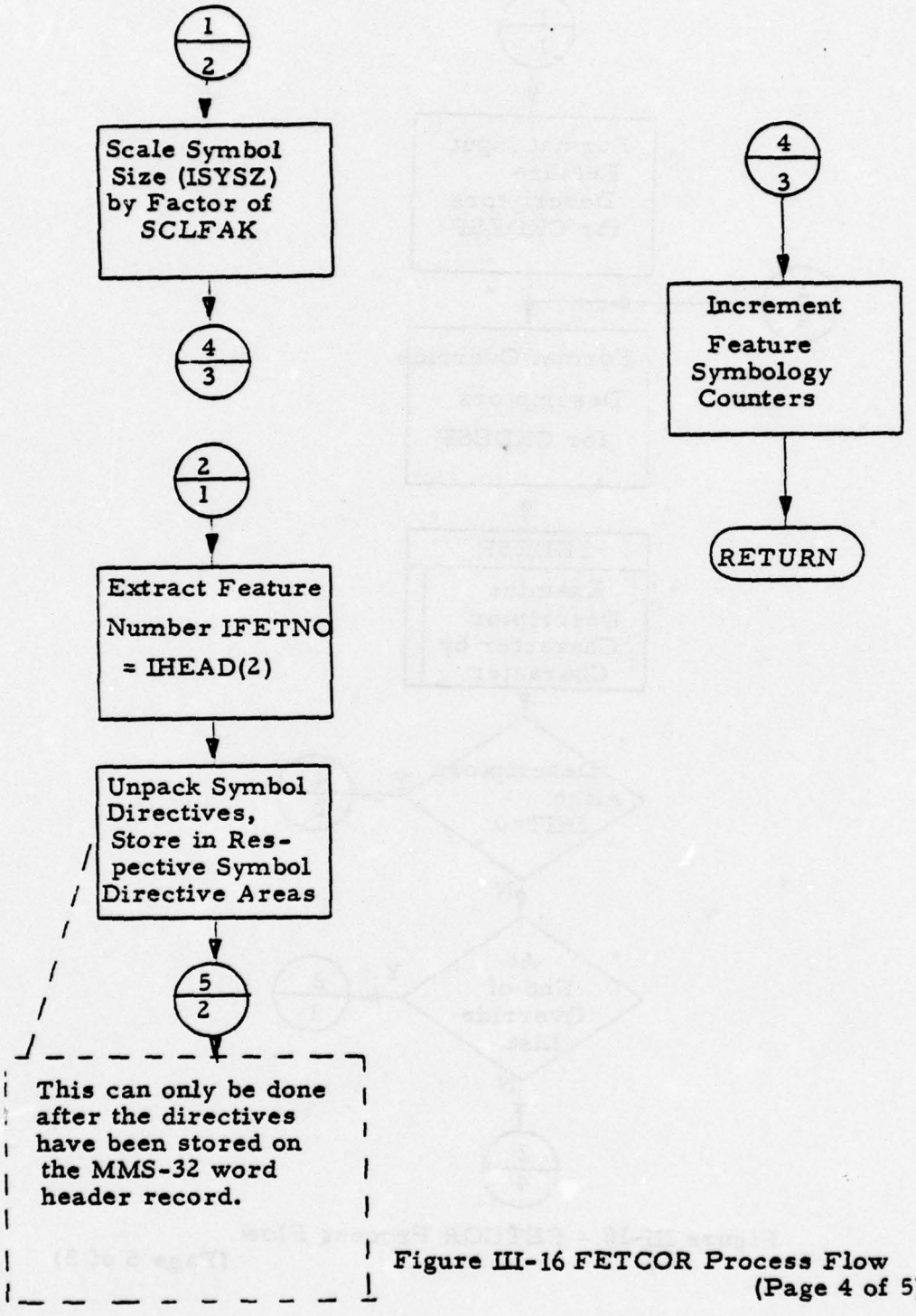



Figure III-16 FETCOR Process Flow
(Page 4 of 5)

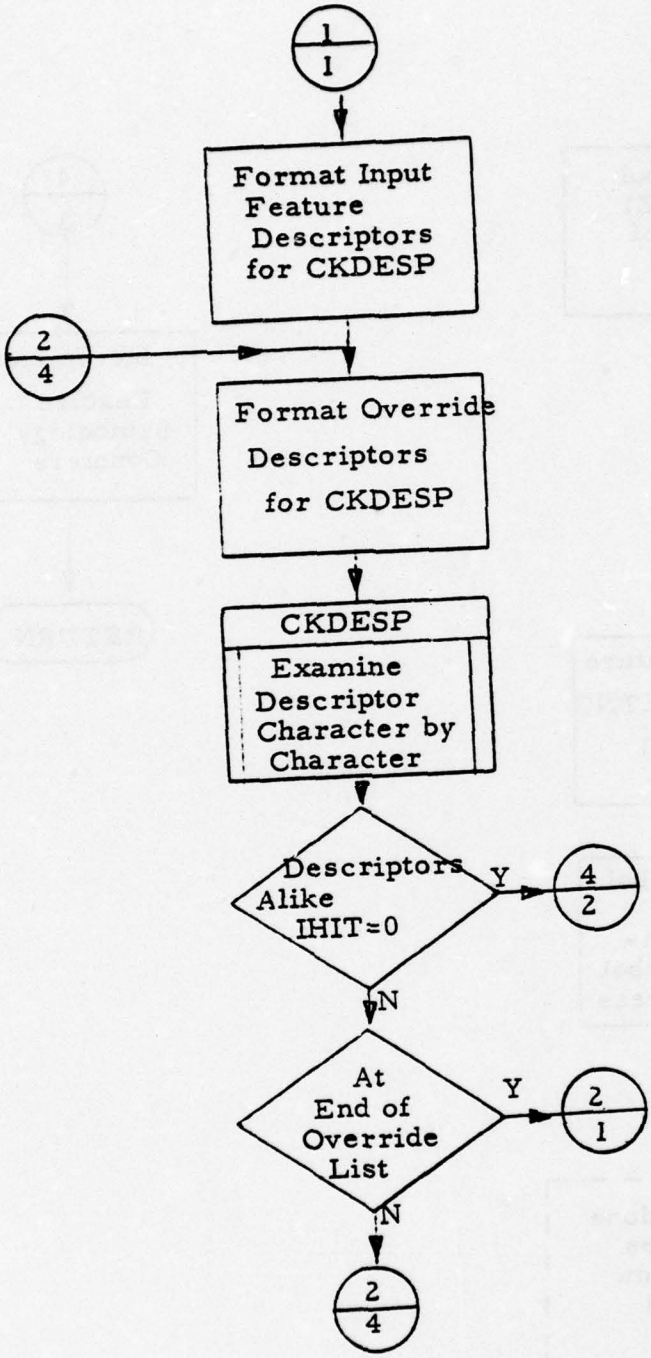



Figure III-16 - FETCOR Process Flow (Page 5 of 5)

## O. CKDESP

### 1. Functional Description

Subroutine CKDESP's primary processing function is to examine input header descriptors against specification descriptors or override descriptors with a resultant of a flag indicating a match or no match. A secondary function is to treat specification or override descriptor characters containing a field data X as a match against the corresponding input header descriptor character.

### 2. Computer Definition

#### a. Core Memory Used

175 octal words

#### b. Peripheral Equipment

Not applicable.

### 3. Program Description

#### a. Calling Routine

FETCOR

#### b. Subroutines Used

None

#### c. Input

Input to Subroutine CKDESP consists of data found in arrays mnemonics ICLS and IDESOV. Array ICLS contains the input header descriptor data while array IDESOV contains specifications or override descriptor data.

#### d. Output

Output from Subroutine CKDESP is simply a flag, mnemonic name IHIT, indicating whether a match was or was not made.

e. Processing Methodology

Entry to Fortran Subroutine CKDESP is made via Subroutine FETCOR. Figure III-17 depicts the process flow of CKDESP. Upon entry, the match no match flag, mnemonic IHIT, is cleared with the index J (word within arrays pointer) being set to one. The input header descriptor and specification or override descriptor data, array ICLS and IDESOV respectively, are indexed by J, moving the J th word to temporary mnemonic ICLSS and IOUR. Index I (character within word pointer) is then set to one. Mnemonic IOUR's $I^{th}$ character is masked, right justified and interrogated. If found to contain a fiel data "X" (octal 35) process control is passed to the next character within mnemonic IOUR. If not a fieldata "X" mnemonic ICLSS corresponding character (I th character) is masked and right justified. The two above masked characters are then examined and if found to be different, output flag IHIT is set to one and process control returned to the calling routine. If the masked characters are found to be alike, index I is examined and if found to be less than or equal to five, is incremented by one with process control being returned to the above character masking. If index I is greater than five, index J is interrogated. When found to be less than three, J is incremented by one with process control being returned to the aforementioned moving of the J th words from ICLS and IDESOV arrays to temporary locations. When index J is found to be greater than or equal to three, process control is returned to the calling routine FETCOR.

f. Calling Sequence

Call CKDESP (ICLS, IDESOV, IHIT)

ICLS - three word array containing input header descriptor data.

IDESOV - three word array containing specification or override descriptor data.

IHIT - flag indicating match or no match

IHIT = 0 match

IHIT = 1 no match.

g. Major Algorithms

None

4. Program Constants and Variables

MASKX - mnemonics is containing fieldata "X" (octal 35)

5. Error Conditions

None

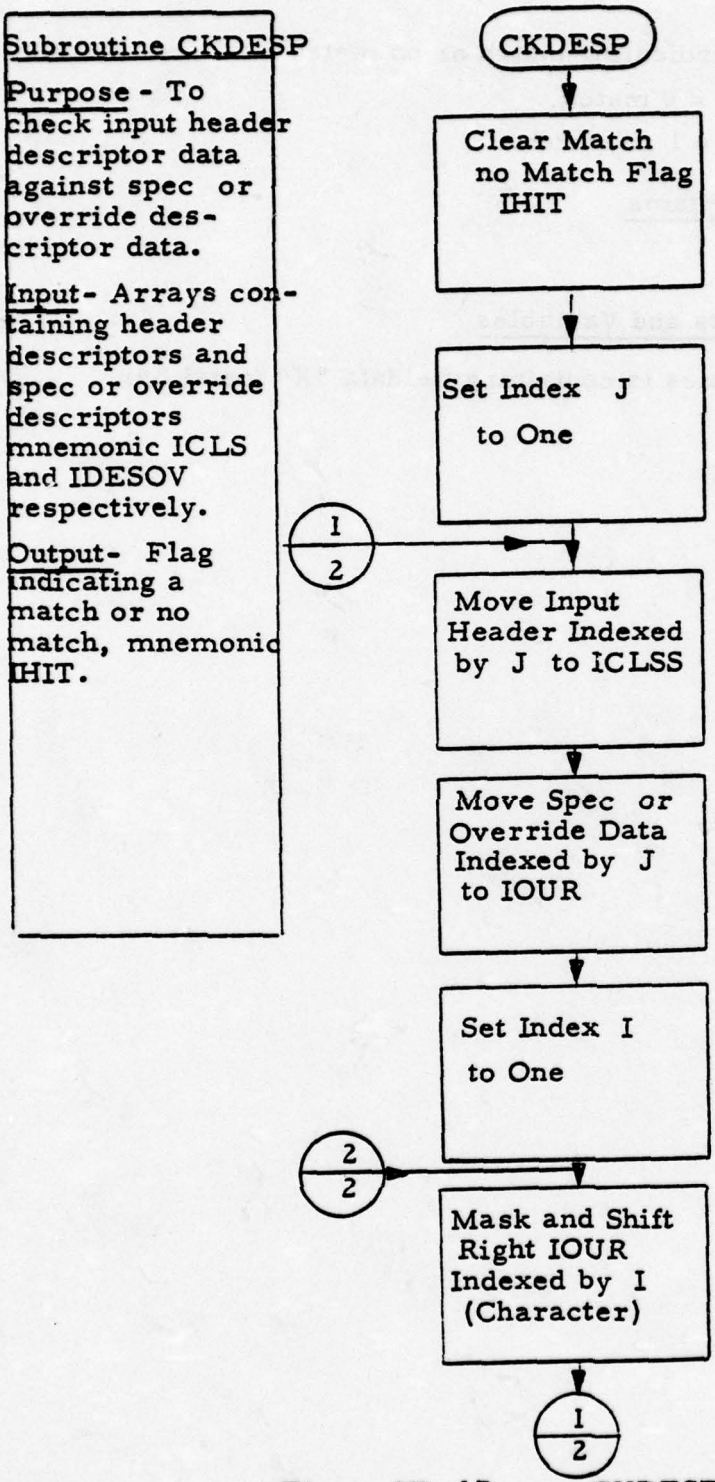



Figure III- 17 - CKDESP Process Flow
(Page 1 of 2)

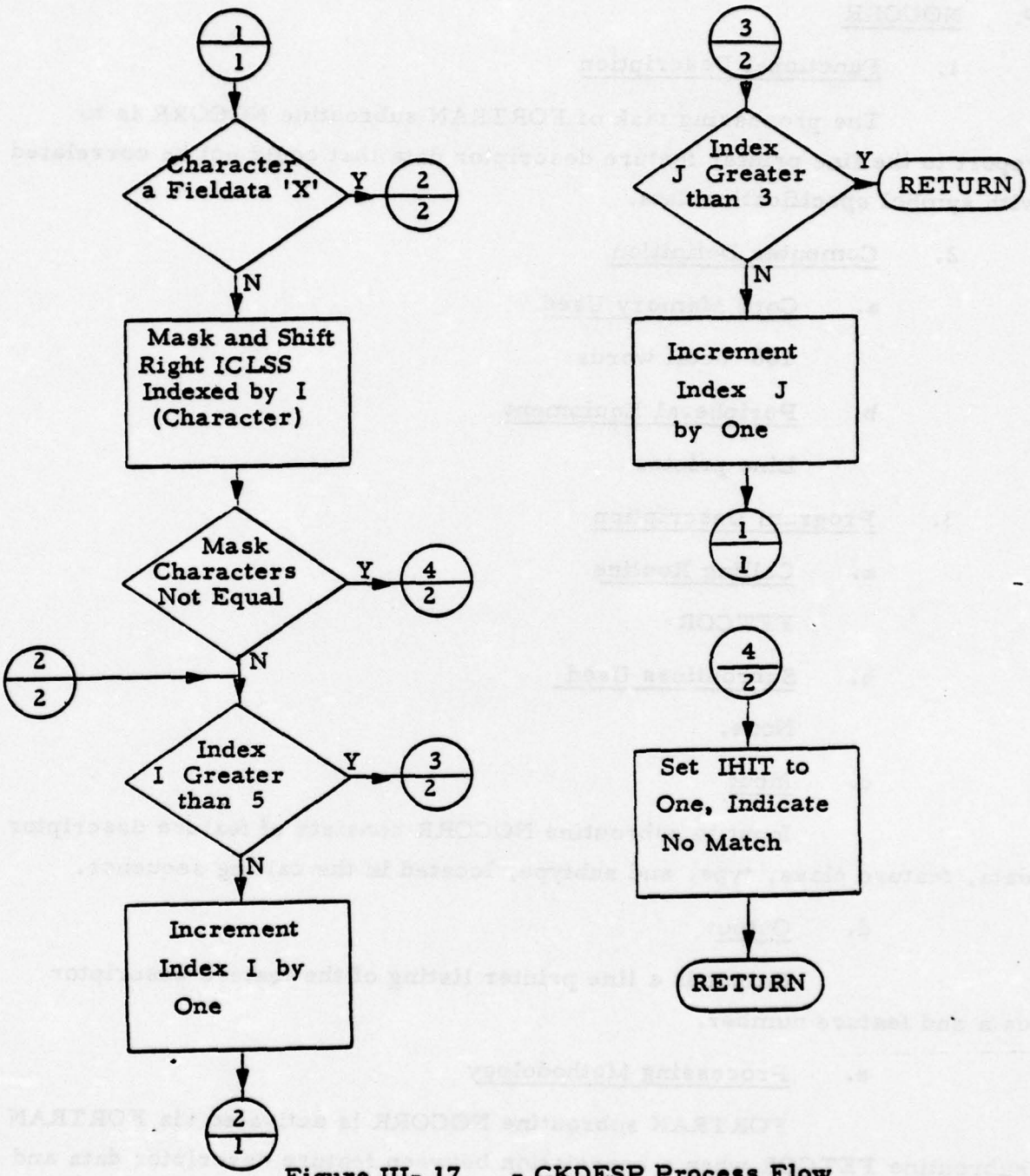



Figure III- 17 - CKDESP Process Flow
(Page 2 of 2)

## P. NOCORR

### 1. Functional Description

The processing task of FORTRAN subroutine NOCORR is to report to the line printer feature descriptor data that could not be correlated with symbol specification data.

### 2. Computer Definition

#### a. Core Memory Used

166 octal words

#### b. Peripheral Equipment

Line printer

### 3. Program Description

#### a. Calling Routine

FETCOR

#### b. Subroutines Used

None.

#### c. Input

Input to subroutine NOCORR consists of feature descriptor data, feature class, type, and subtype, located in the calling sequence.

#### d. Output

Output is a line printer listing of the feature descriptor data and feature number.

#### e. Processing Methodology

FORTRAN subroutine NOCORR is activated via FORTRAN subroutine FETCOR when a correlation between feature descriptor data and symbol specification data can not be associated. Upon receiving process control the feature descriptor data and feature number passed to NOCORR in the calling sequence is listed to the line printer. Process control is then returned to the calling routine FETCOR.

f. Calling Sequence

Call NOCORR (ICL, IFTNUM)

ICL - array containing the feature descriptor data

IFTNUM - feature number

g. Major Algorithms

None.

4. Program Constants and Variables

N/A

5. Error Conditions

None.

Q. SMOOTH

1. Functional Description

As a FORTRAN subroutine, SMOOTH performs cleaning, data reduction and smoothing processes on line center data prior to application of graphic symbols.

2. Computer Definition

a. Core Memory Used

2722 octal words

b. Peripheral Equipment

None

3. Program Description

a. Calling Routines

MONITR

b. Subroutines Used

SQRT
BACKUP
ATAN

c. Input

Inputs to subroutine SMOOTH consist of data, contained in common areas C2 (feature line center data) and C6 (parameters and variables).

d. Output

Outputs from subroutine SMOOTH consist of line center data in common area C2 which has been smoothed. Also, output consists of tally summary reports in common area C7.

e. Processing Methodology

When first called, the subroutine SMOOTH tests whether or not the input file (parameter IFILE) is a LIS table coordinate file. If the LIS file is determined as the input file, SMOOTH proceeds to convert the input maximum distance, minimum distance, and slope distance to metric units and stores the converted values in the appointed common area. After the conversion, or if LIS table coordinate file is not the input file, SMOOTH sets a flag (IFIRST=2) to avoid subsequent testing of the input file parameter on succeeding entries.

The subroutine SMOOTH then determines if the input line center feature contains more than two data points. When the input line center feature contains at most two points, no cleaning process is performed. By interrogating the smooth option selected by the user (mnemonic ISMOTH), subroutine SMOOTH proceeds with performing the option selected through the data as follows:

- Smooth Option 0

  Subroutine SMOOTH computes only the distance between two successive points and moves the distance value squared to the data buffer.

- Smooth Option 1

  Subroutine SMOOTH calculates the distances between two successive points at which time a test is made to determine if the distance is greater than the minimum distance. Points, other than special points, which are determined to be within the minimum distance, are deleted from the data list.

- Smooth Option 2

  When subroutine SMOOTH is called with option 2, the distances between successive points is first computed. After the distances are determined, SMOOTH steps through the data list in buffer two, computes the midpoints between sequential points, computes and examines the distances, and stores the midpoints with their

distances into buffer one. The preceding operation is not performed when two points are determined to be point-point or when a special point is encountered. Point-point data and special points are directly stored in buffer one.

- Smooth Option 3

  On a call to subroutine SMOOTH with option 3, the cleaning process is performed to delete points which are not in continuity of the data list. SMOOTH transverses through the data list in buffer two, tests the continuity of four sequential data points by a call to BACKUP, deletes the third point which does not conform with the data, and stores the clean data points with the computed sequential distances greater than the minimum distance in buffer one. While deleting points which do not conform with the data, SMOOTH will maintain all points which are flagged as special points and end points of point-point data (smooth option 1 is performed before the above).

- Smooth Option 4

  With option 4, the subroutine SMOOTH will first perform option 2, then follow with option 3, When option 3 is performed, the data points have already been transferred from buffer two to buffer one in the operation of option 2.

- Smooth Option 5

  When subroutine SMOOTH encounters option 5, SMOOTH will first proceed with processing the input through option 2 (see the above). After completing option 2, the smooth data in buffer one will be processed again. That is, the midpoints are computed with their sequential distances and stored in buffer one, while keeping those data points which are flagged as special or end points of point-point data.

f. Calling Sequence

Call SMOOTH

g. Major Algorithms

- Smooth Option 0

Let $(x_i, y_i)$ and $(x_{i+1}, y_{i+1})$ be coordinates of adjacent points, and let $d_{i+1}$ be the distance from the i th point to the i+1 th point. Option 0 only computes:

$$d_{i+1} = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$$

- Smooth Option 1

Given $(x_i, y_i)$, $(x_{i+1}, y_{i+1})$, and $d_i$ as in smooth option 1, and given IMINDT as the minimum resolution

then,

if $d_i < \text{IMINDT}$,

then $x_i = x_{i+1}$

$y_i = y_{i+1}$

and

$$d_i = \sqrt{(x_{i+1} - x_{i-1})^2 + (y_{i+1} - y_{i-1})^2} .$$

- Smooth Option 2

Let $(x_i, y_i)$ and $(x_{i+1}, y_{i+1})$ be adjacent points, then the midpoint $(\bar{x}_{i+1}, \bar{y}_{i+1})$ will be computed as

$$\bar{x}_{i+1} = 1/2 (x_i + x_{i+1})$$

$$\bar{y}_{i+1} = 1/2 (y_i + y_{i+1})$$

and

$\bar{x}_1 = x_1, \quad \bar{y}_1 = y_1$

with

$\bar{x}_{n+1} = x_n, \quad \bar{y}_{n+1} = y_n$, where n is the last point of the data list, a special point, or the first point of point-point.

- Smooth Option 3

Let $(x_{i-2}, y_{i-2})$, $(x_{i-1}, y_{i-1})$, and $(x_i, y_i)$ be three successive data points in buffer one,

and let $(x_j, y_j)$ be a sequential point following in buffer two, such that, the j-1 point in buffer two is the ith point in buffer one.

On return from a call to

BACKUP $(x_{i-2}, y_{i-2}, x_{i-1}, y_{i-1}, x_i, y_i, x_j, y_j, i, j)$,

if $i \neq j$, the point $(x_j, y_j)$ is moved into location i in the first buffer,

or

if $i = j$, the point $(x_j, y_j)$ is moved into location i+1 in the first buffer.

(Refer to BACKUP, Section R).

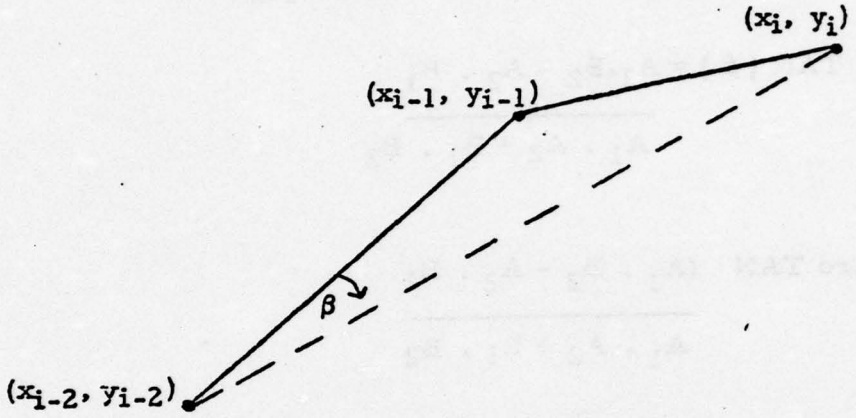
- Smooth Option 4

Option 4 is basically an iteration of option 2, then followed by option 3 with points in buffer one. (Refer to options 2 and 5).

- Smooth Option 5

  Two iterations of option 2 are performed: the first iteration with points in buffer two, and the second iteration with points in buffer one.

- Smooth Option 6




Let $(x_{i-2}, y_{i-2})$, $(x_{i-1}, y_{i-1})$ and $(x_i, y_i)$ be three successive data points in buffer two.

Now let

(1) $A_1 x + B_1 y + C_1 = 0$ be the equation of the line formed by the points $(x_{i-2}, y_{i-2})$ and $(x_{i-1}, y_{i-1})$, and let:

(2) $A_2 x + B_2 y + C_2 = 0$ be the equation of the line formed by the points $(x_{i-2}, y_{i-2})$ and $(x_i, y_i)$.

From the equation of a straight line formed by two points, it can be determined that for the first equation (1) we have $(y_{i-1} - y_{i-2}) X - (x_{i-1} - x_{i-2}) Y + (x_{i-1} - x_{i-2}) y_{i-2} - (y_{i-1} - y_{i-2}) x_{i-2}) = 0$.

Therefore,

$$A_1 = y_{i-1} - y_{i-2}$$
$$B_1 = x_{i-2} - x_{i-1}$$
$$C_1 = (x_{i-1} - x_{i-2}) y_{i-2} - (y_{i-1} - y_{i-2}) x_{i-1}$$

For equation (2), we can determine that:

$$(y_i - y_{i-2})\,x - (x_i - x_{i-2})\,y + (x_i - x_{i-2})\,y_{i-2} - (y_i - y_{i-2})\,x_{i-2} = 0.$$

Therefore,

$$A_2 = y_i - y_{i-2}$$
$$B_2 = x_{i-2} - x_i$$
$$C_2 = (x_i - x_{i-2})\,y_{i-2} - (y_i - y_{i-2})\,x_{i-2} = 0$$

Now: $$\text{TAN}(\beta) = \frac{A_1 \cdot B_2 - A_2 \cdot B_1}{A_1 \cdot A_2 + B_1 \cdot B_2}$$

and,

$$\beta = \text{Arc TAN}\ \frac{(A_1 \cdot B_2 - A_2 \cdot B_1}{A_1 \cdot A_2 + B_1 \cdot B_2}$$

Given that $\alpha$ is the maximum angle of delineation, and if $\beta \leq \alpha$, then: $(x_{i-1}, y_{i-1})$ is deleted.

4. Program Constants and Variables

| | | |
|---|---|---|
| ALPHA | - | maximum angle at delineation allowed with smooth option 6. |
| ICORDX | - | current buffer pointer into IXYZ |
| ID | - | computed (square) distance between two points |
| IMAXDT | - | maximum distance to be considered as trace data |
| IMAXPT | - | maximum distance (squared) to be considered as trace data |
| IMINDT | - | minimum distance . or closest point tolerance |
| IOPT | - | smooth option plus one |
| IPTDLT | - | cumulative total of points deleted by SMOOTH |

| | | |
|---|---|---|
| IPTPRS | - | cumulative total of points processed by SMOOTH |
| IPTPSS | - | cumulative total of points passed by SMOOTH |
| ISMOTH | - | SMOOTH option |
| ISPPTS | - | array containing the special point indexes |
| IX | - | differences in x coordinates |
| IXYZ | - | two-dimensional array containing x, y coordinate points with distance between two successive points |
| IY | - | differences in y coordinates |
| RADI | - | 0.017453 radians for one degree |
| RNINETY | - | 1.570796 radians for ninety degrees |

5. Error Conditions

None.

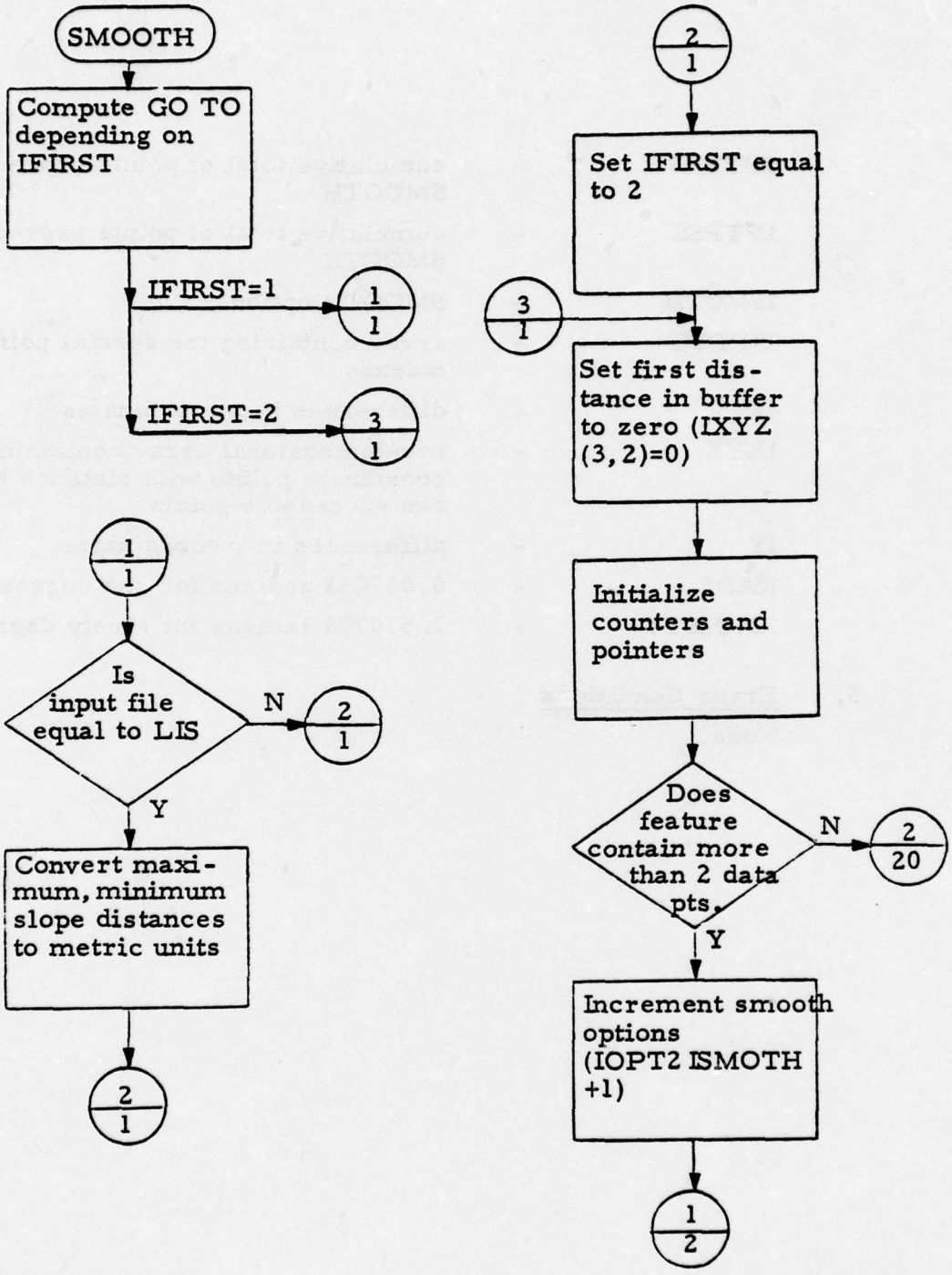



Figure III-18 SMOOTH Process Flow (Page 1 of 20)

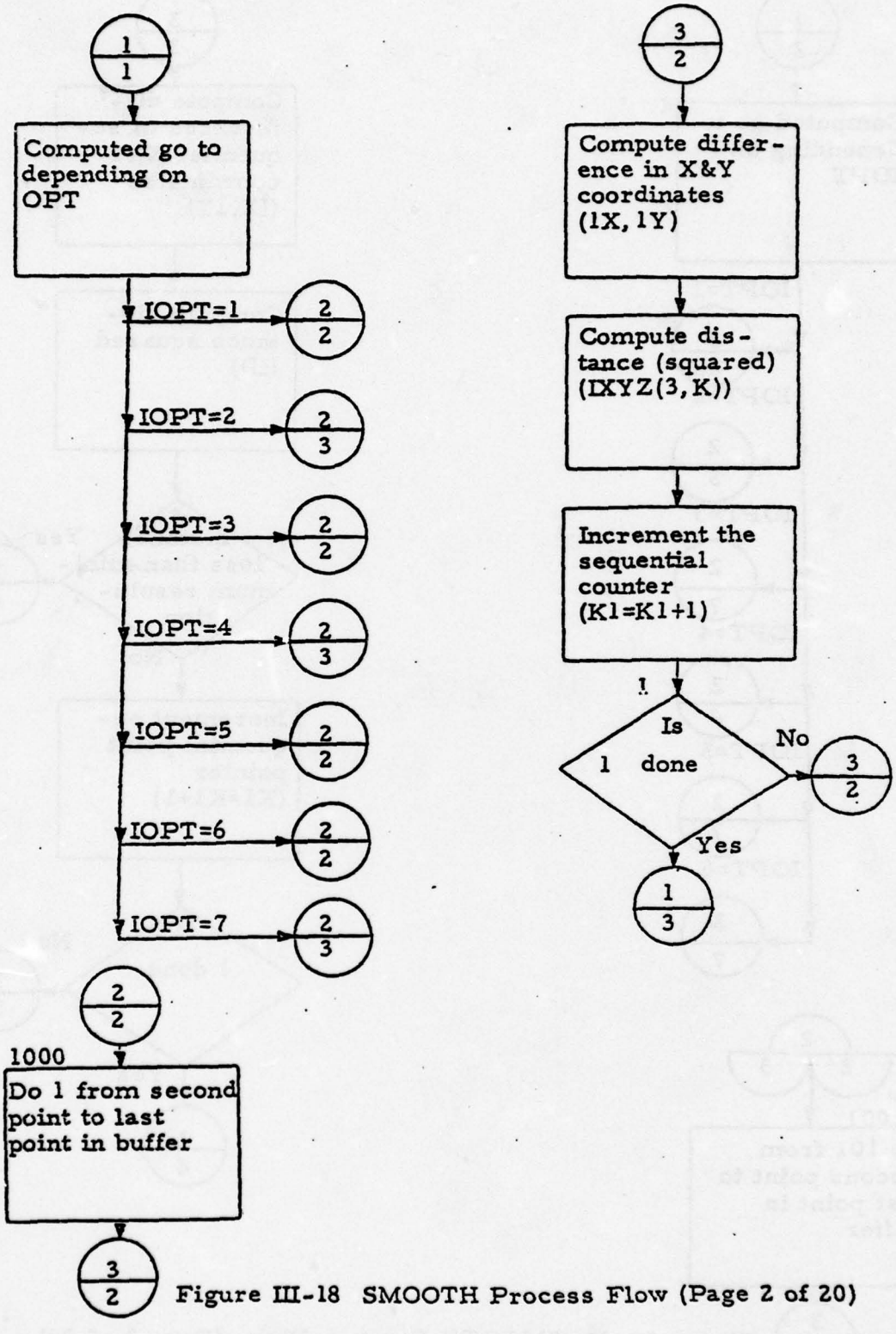



Figure III-18 SMOOTH Process Flow (Page 2 of 20)

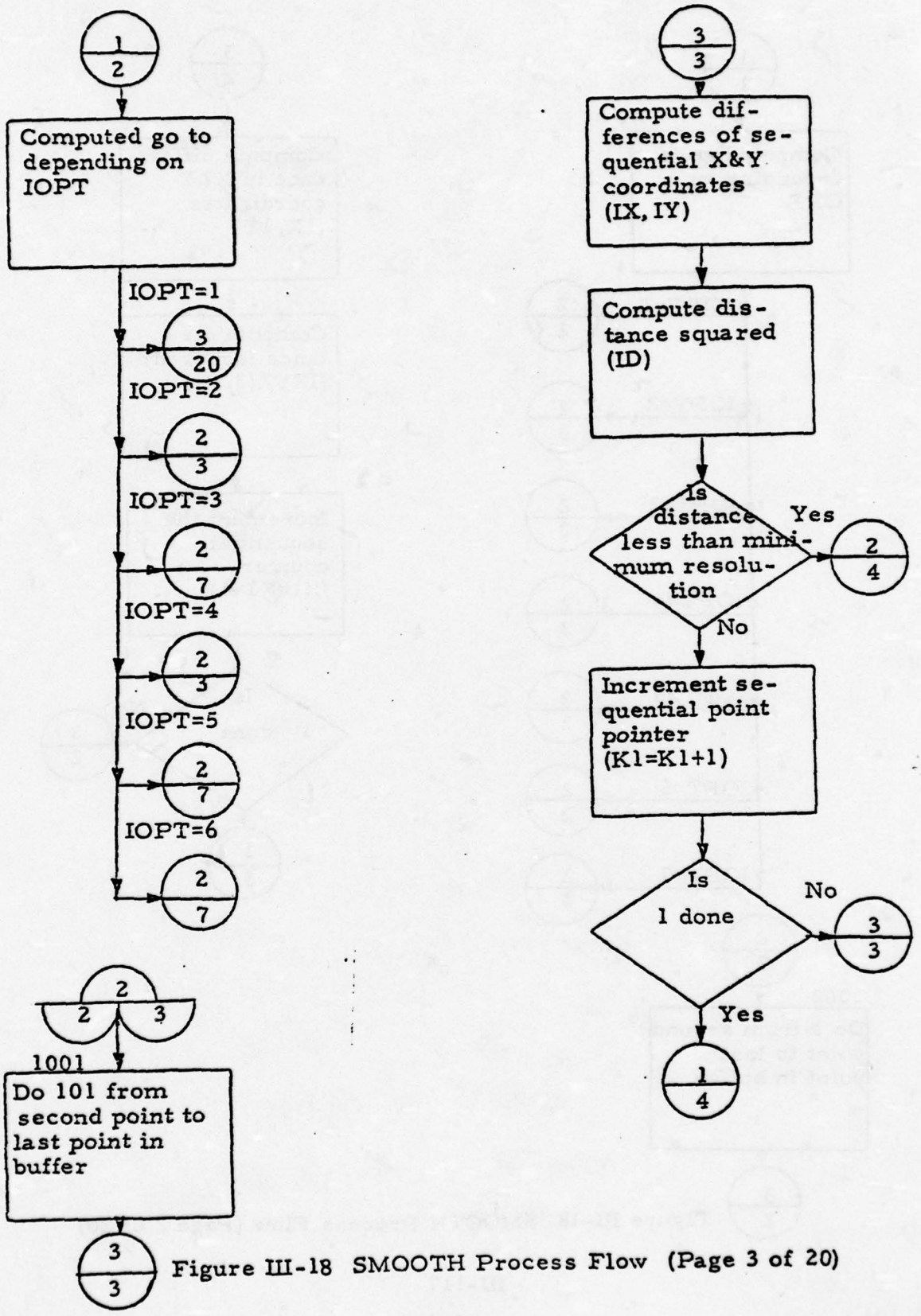



Figure III-18 SMOOTH Process Flow (Page 3 of 20)

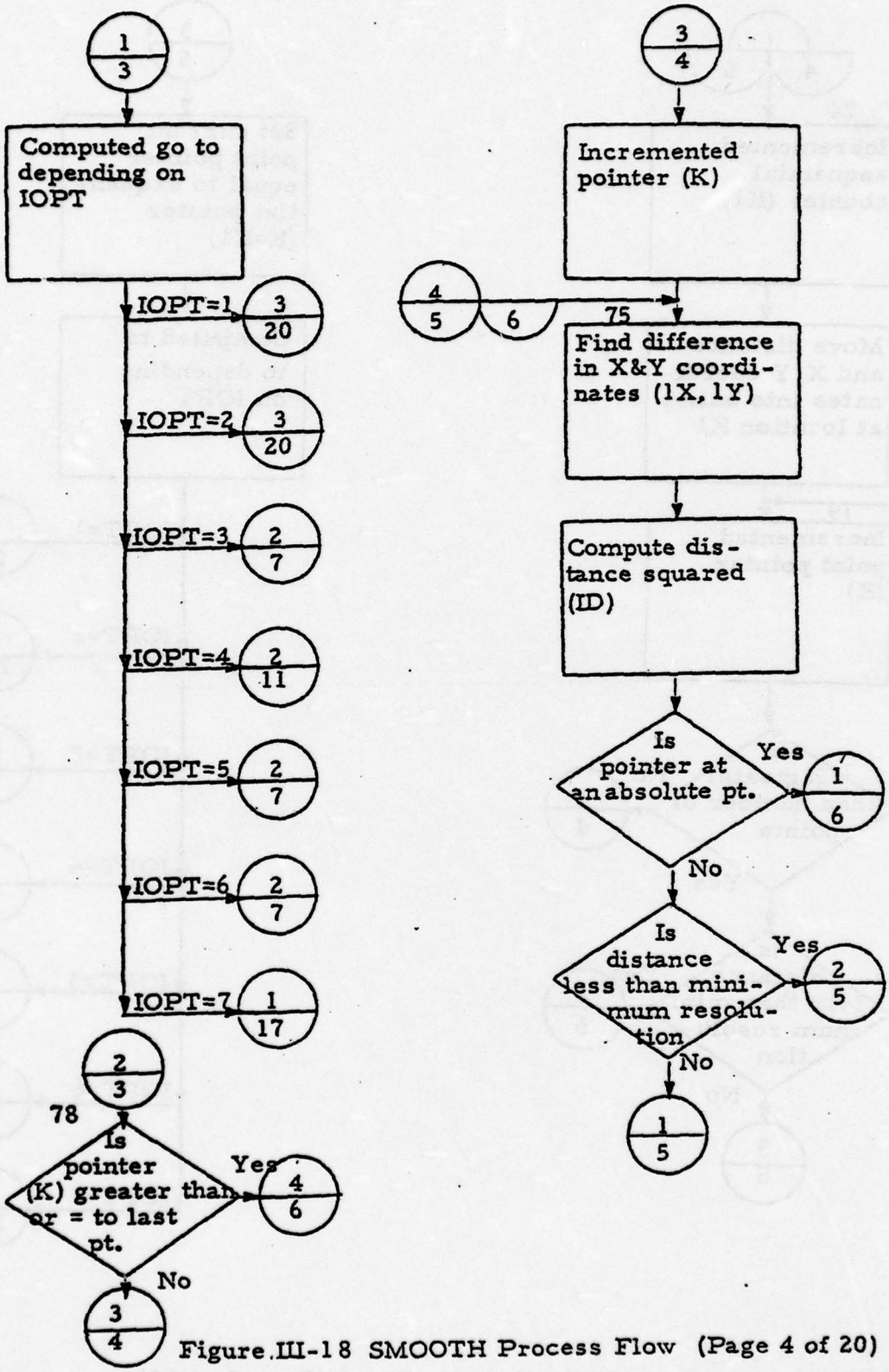



Figure III-18 SMOOTH Process Flow (Page 4 of 20)

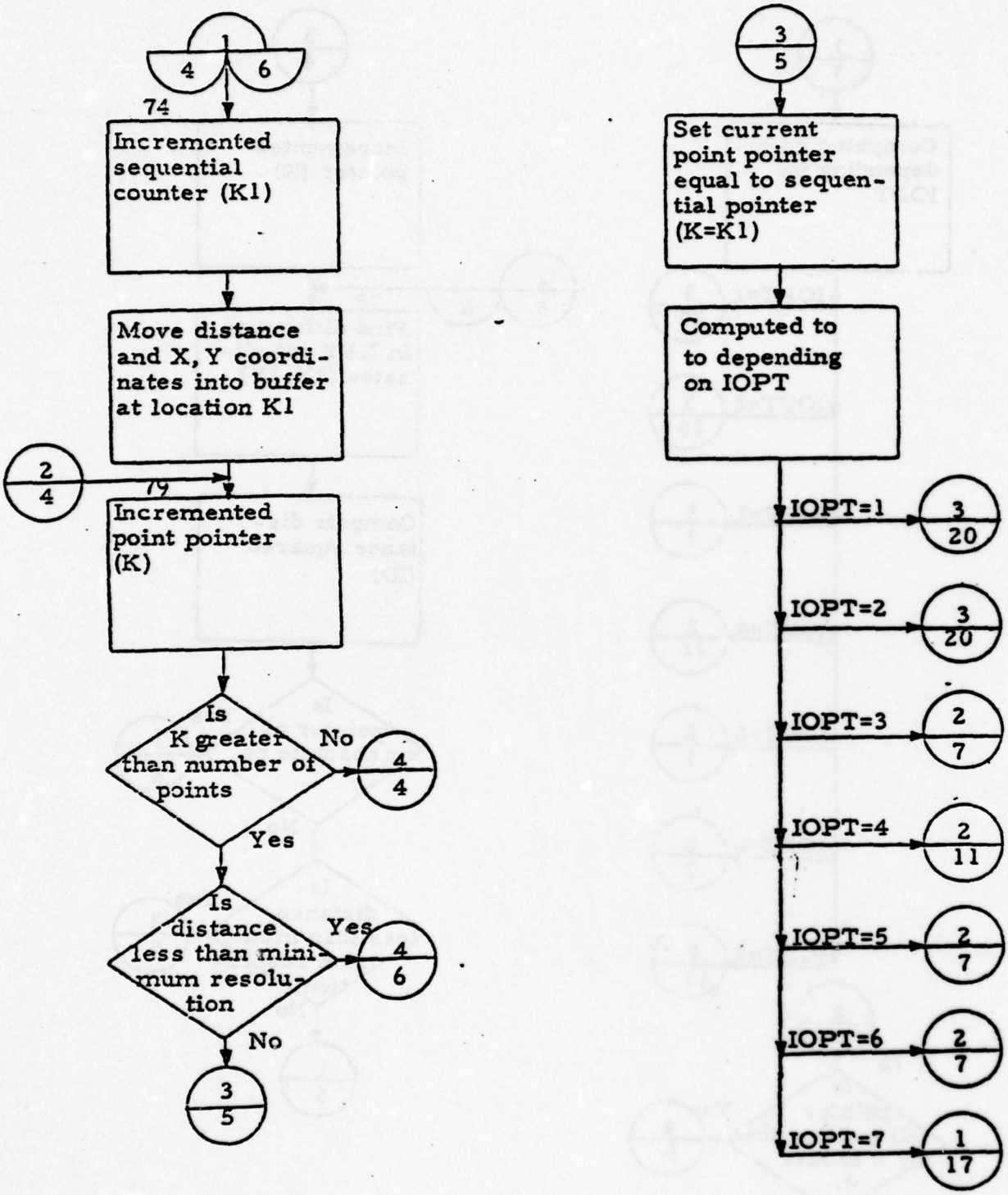



Figure III-18 SMOOTH Process Flow (Page 5 of 20)

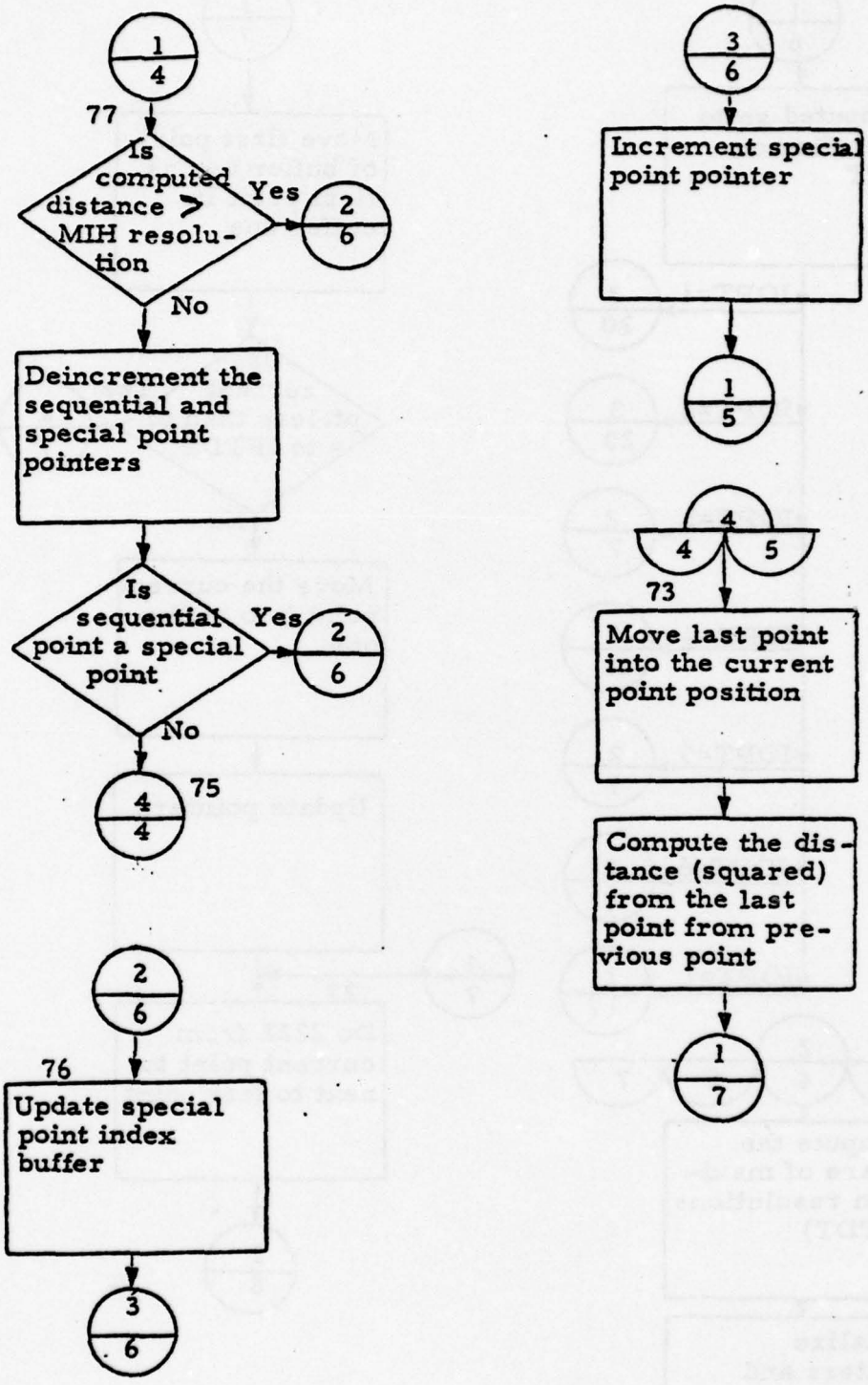



Figure III-18 SMOOTH Process Flow (Page 6 of 20)

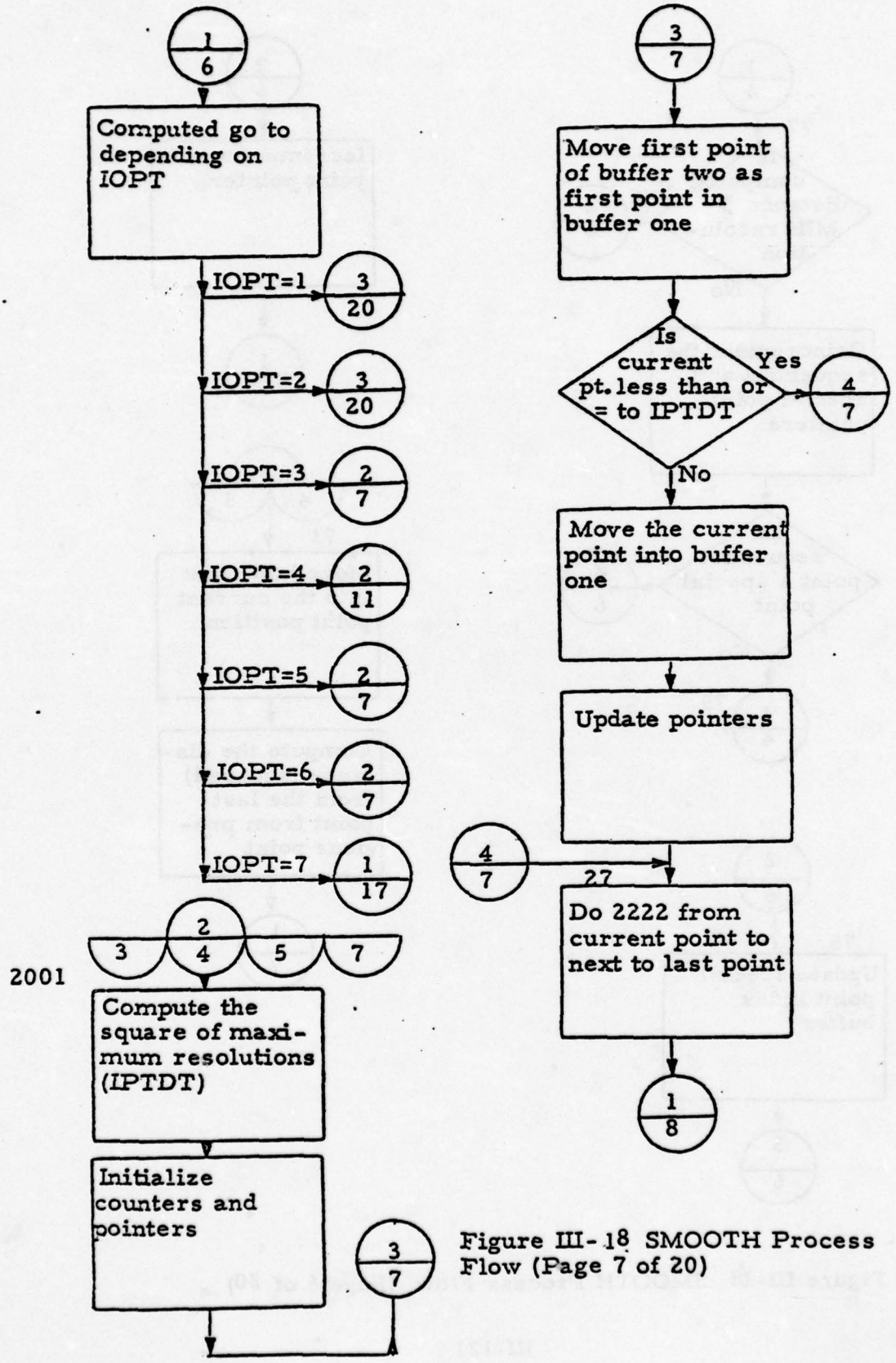



Figure III-18 SMOOTH Process Flow (Page 7 of 20)

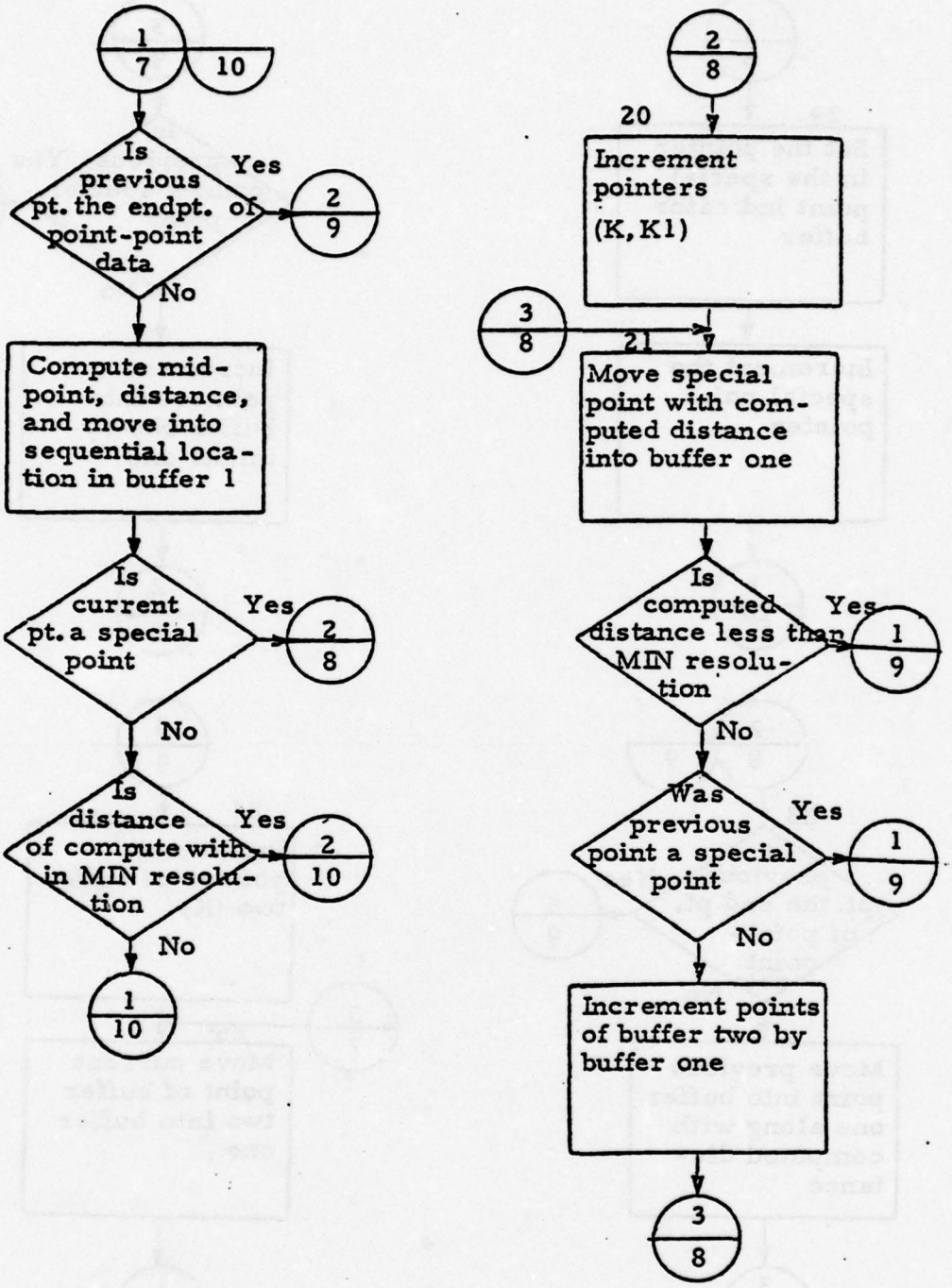



Figure III-18 SMOOTH Process Flow (Page 8 of 20)

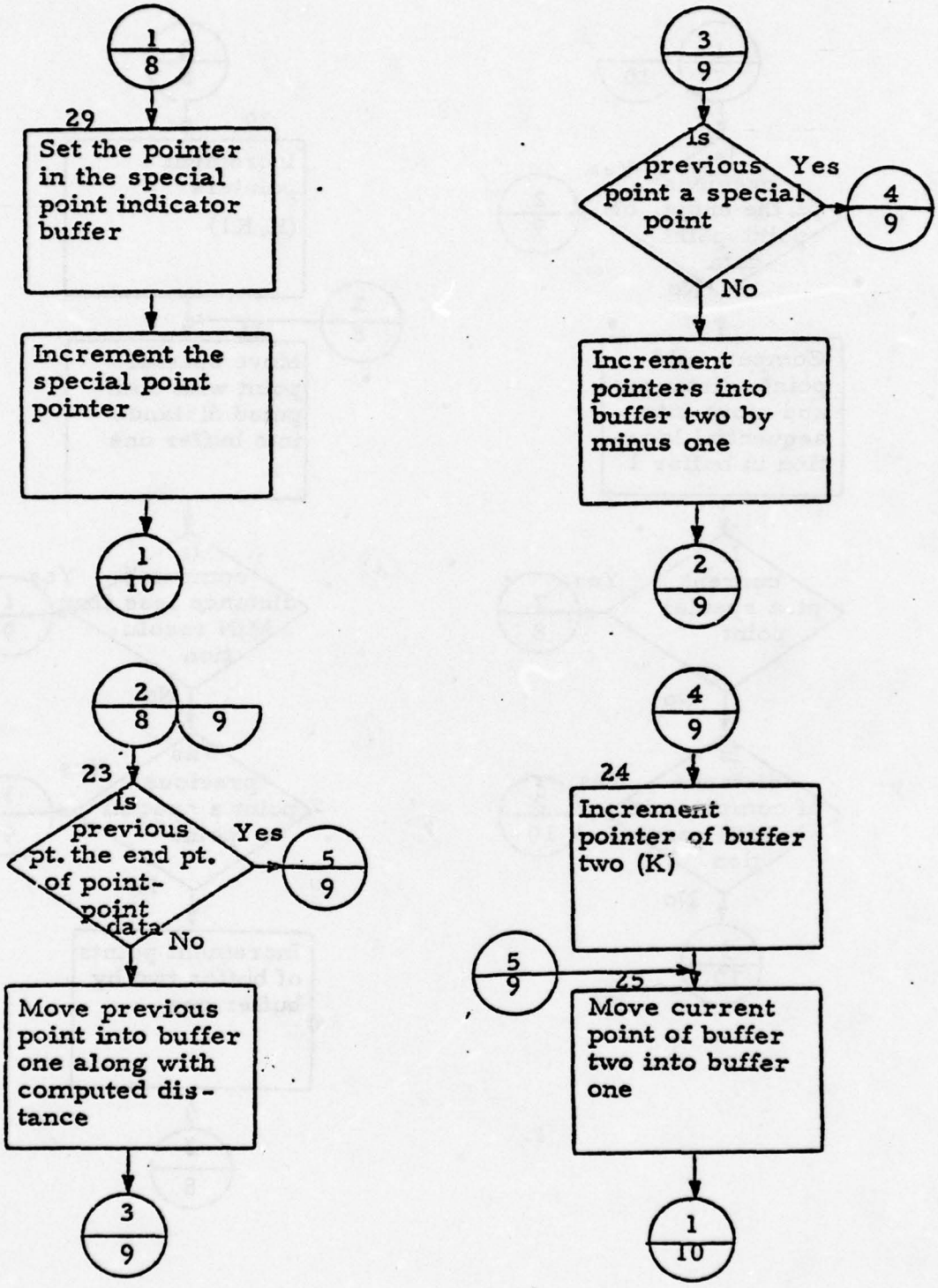



Figure III-18 SMOOTH Process Flow (Page 9 of 20)

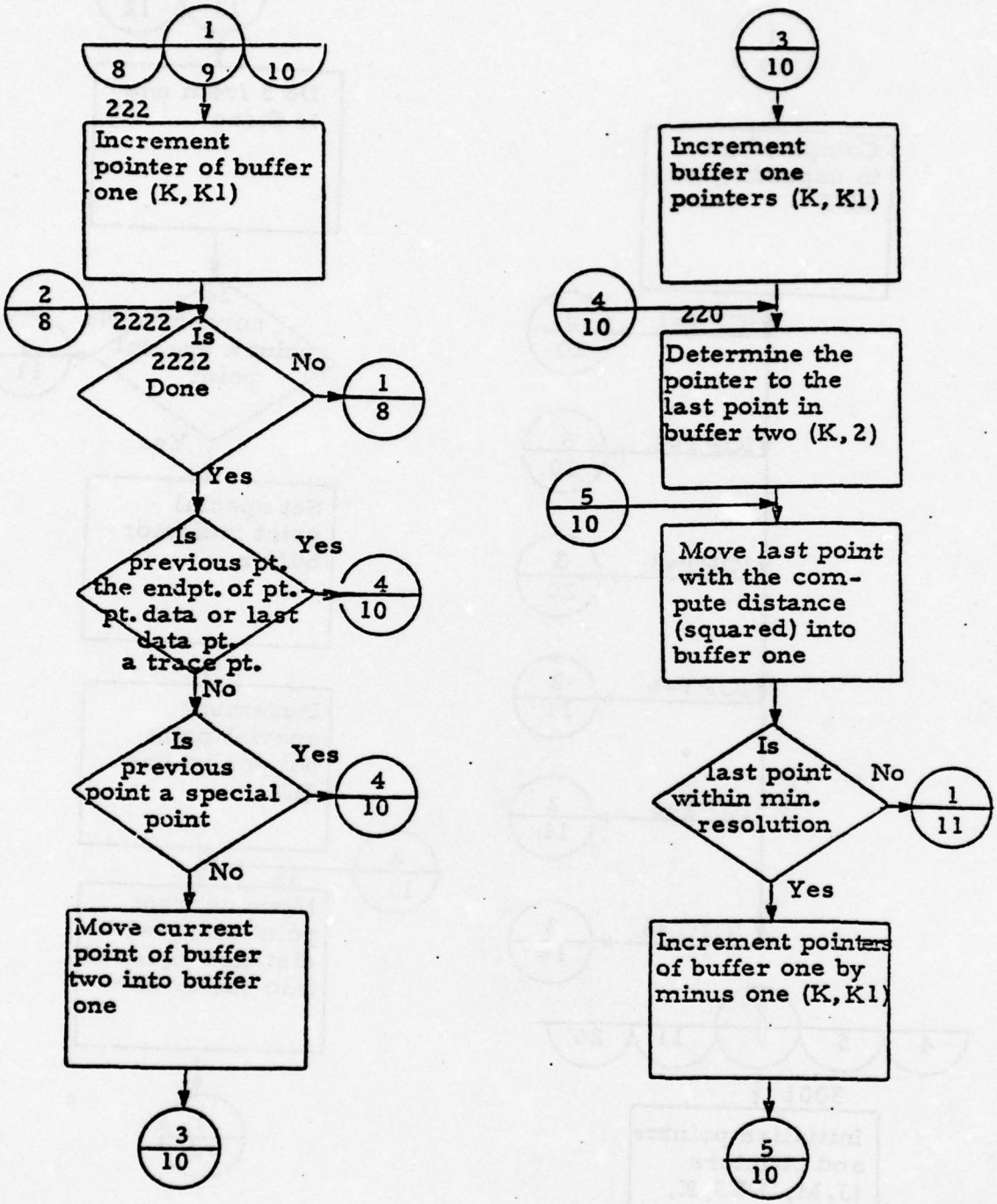



Figure III-18 SMOOTH Process Flow (Page 10 of 20)

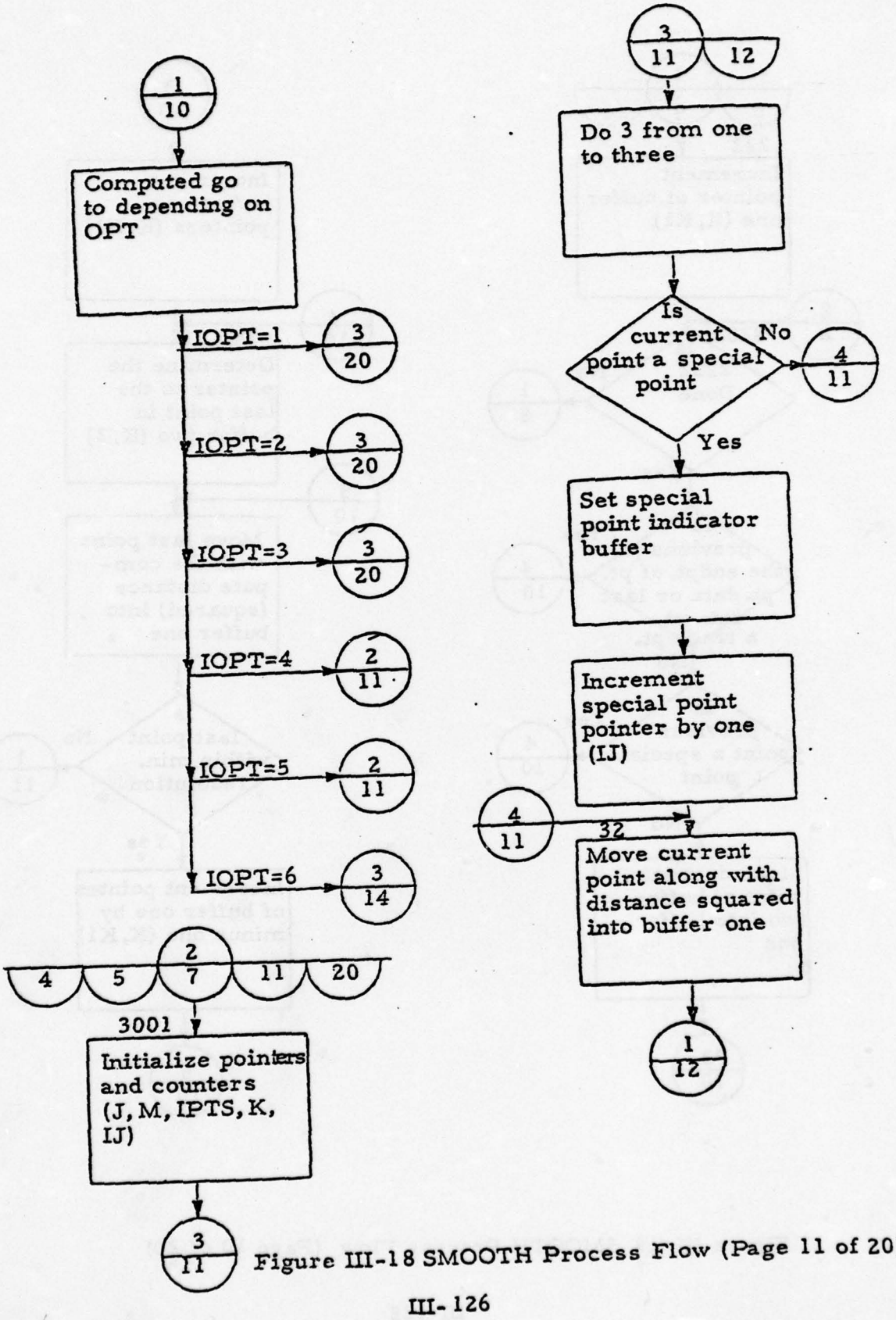



Figure III-18 SMOOTH Process Flow (Page 11 of 20)

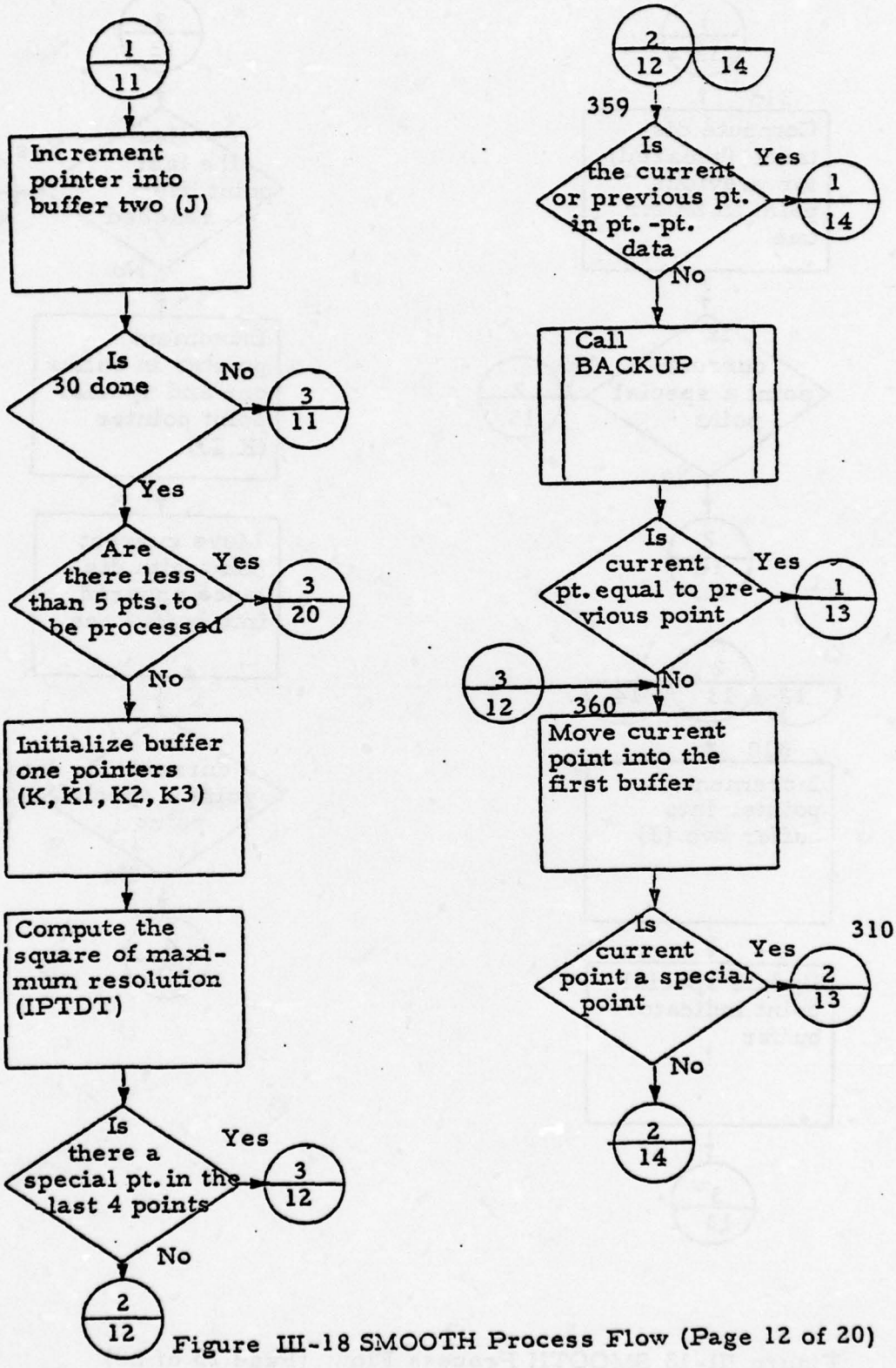



Figure III-18 SMOOTH Process Flow (Page 12 of 20)

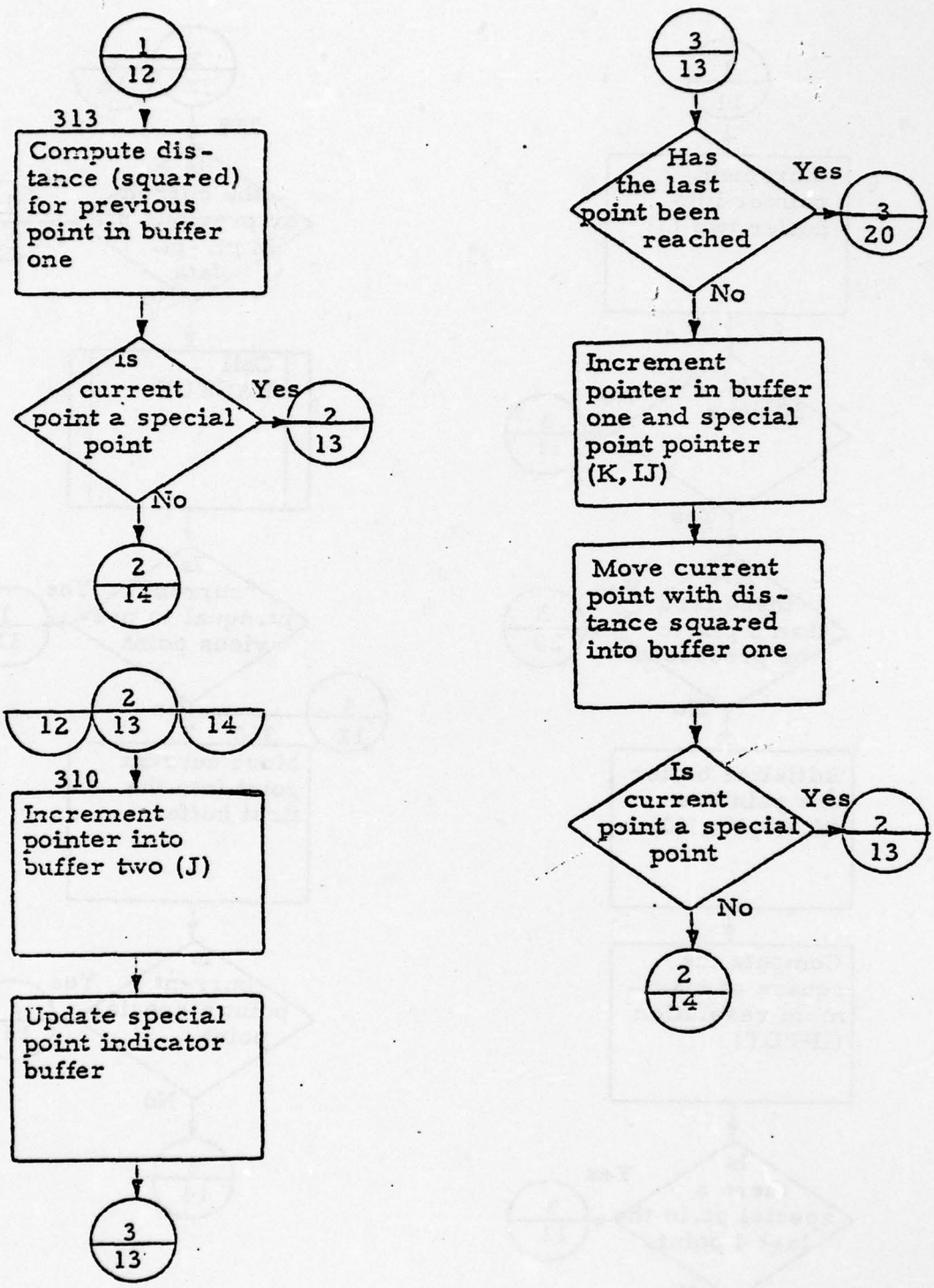



Figure III-18 SMOOTH Process Flow (Page 13 of 20)

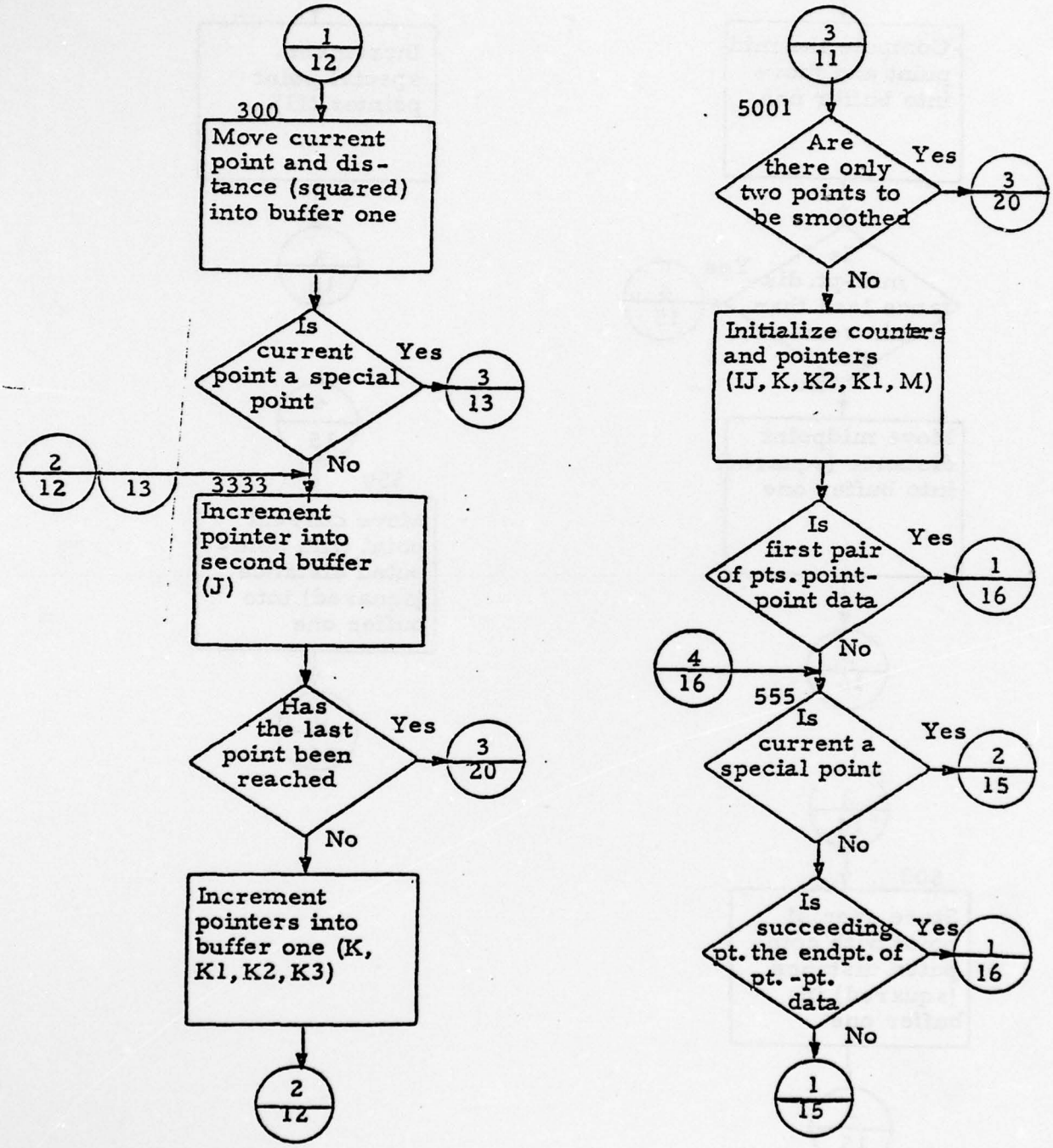



Figure III-18 SMOOTH Process Flow (Page 14 of 20)

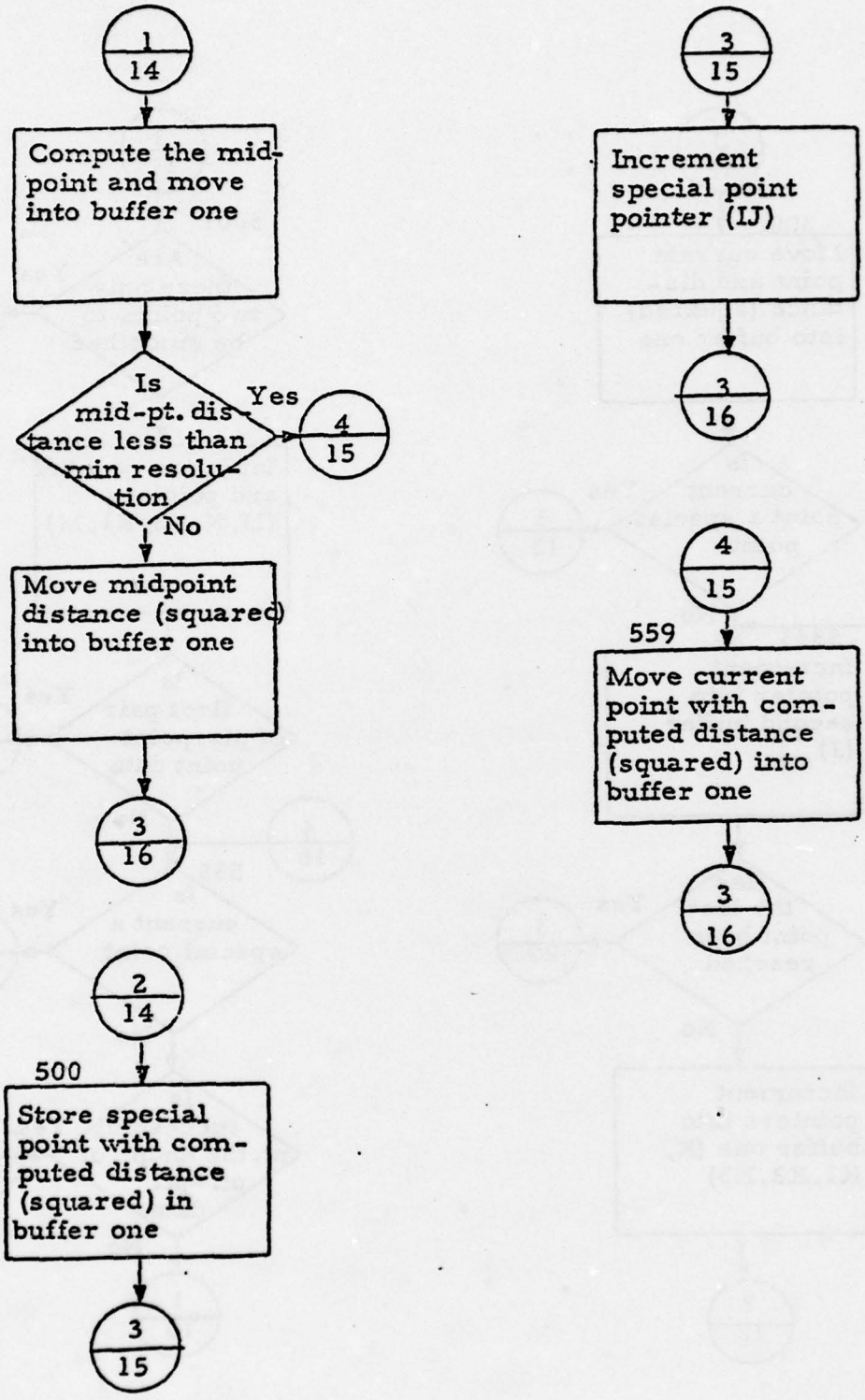



Figure III-18 SMOOTH Process Flow (Page 15 of 20)

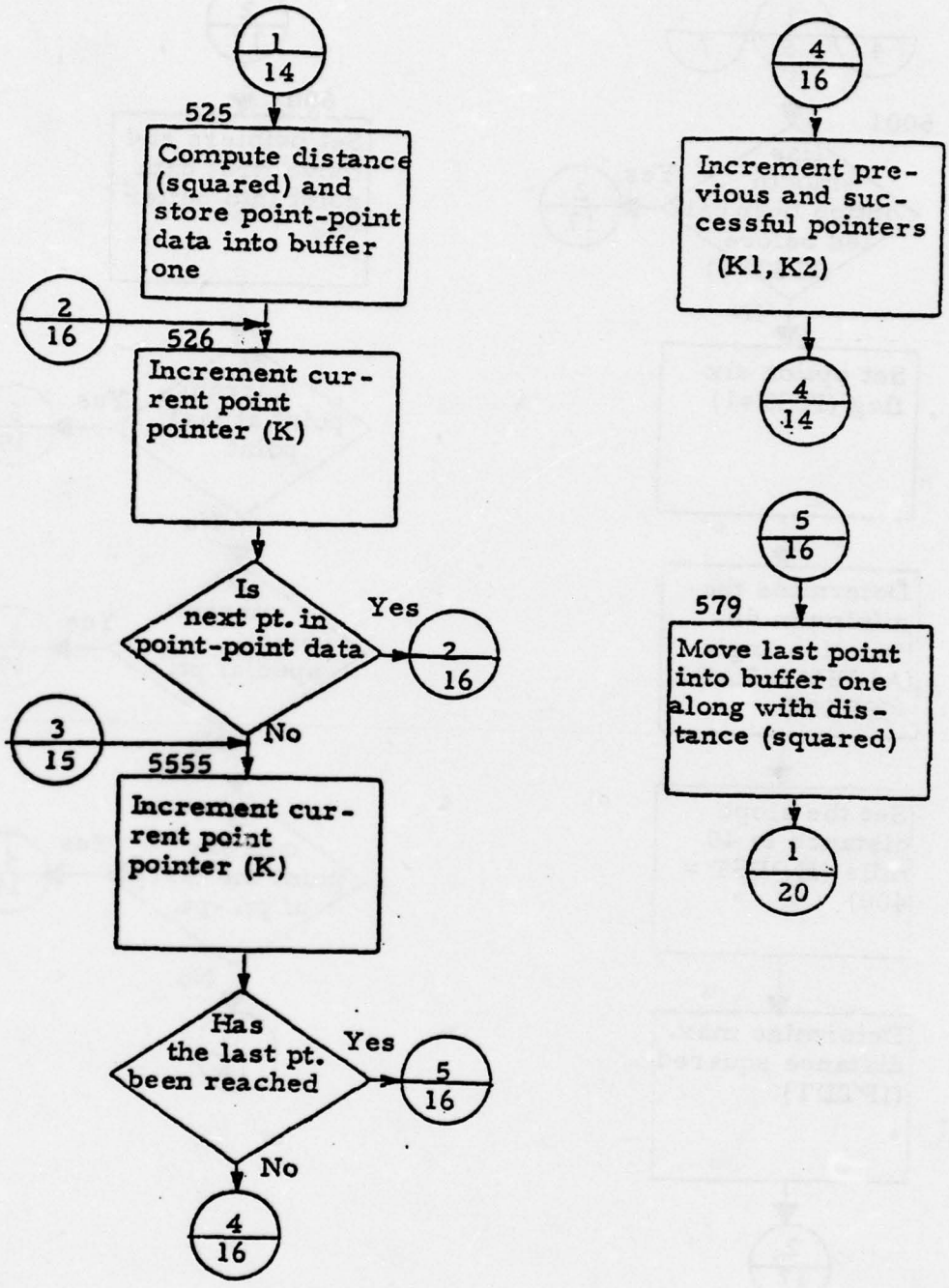



Figure III-18 SMOOTH Process Flow (Page 16 of 20)

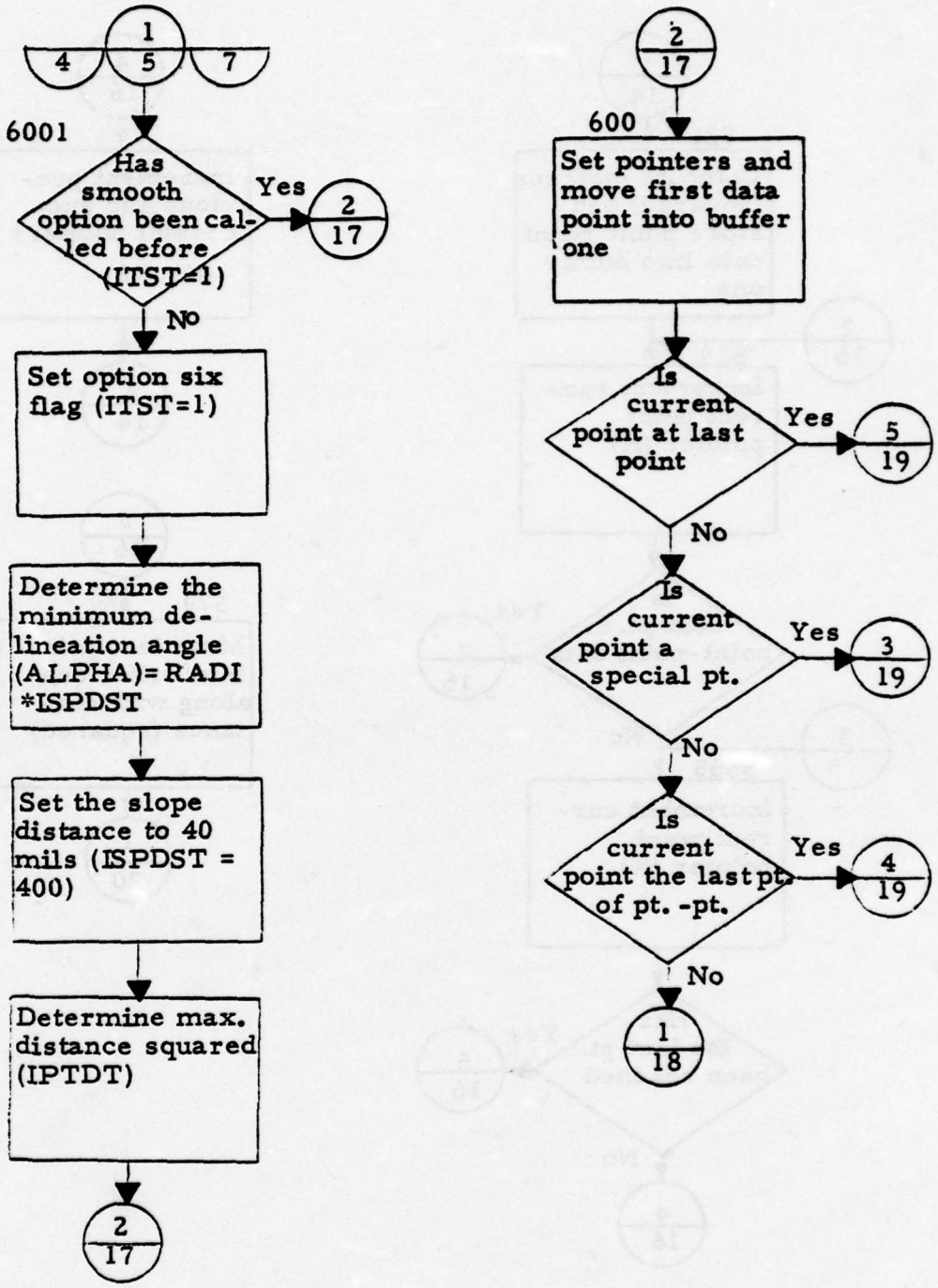



Figure III-18 SMOOTH Process Flow (Page 17 of 20)

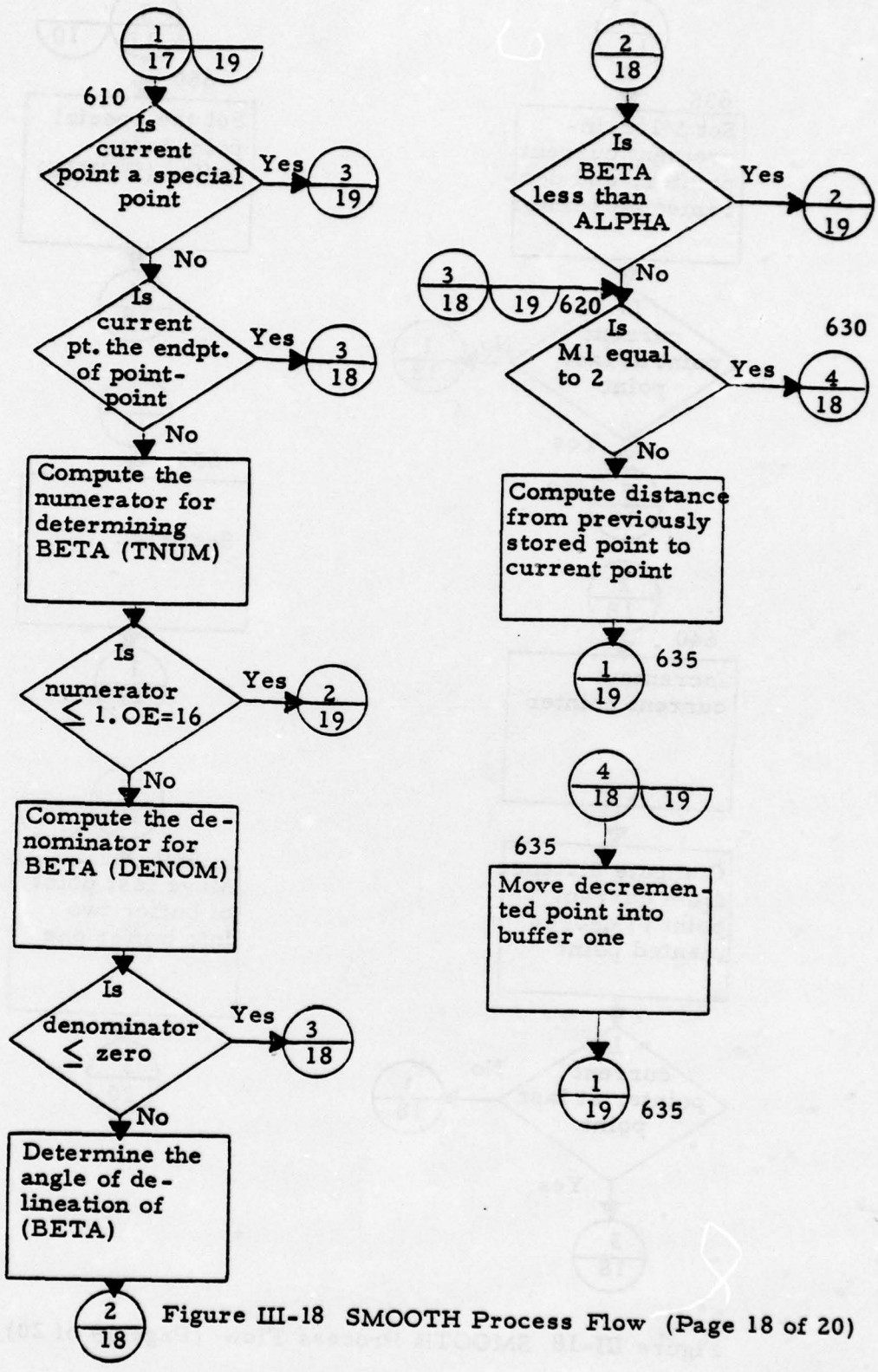



Figure III-18 SMOOTH Process Flow (Page 18 of 20)

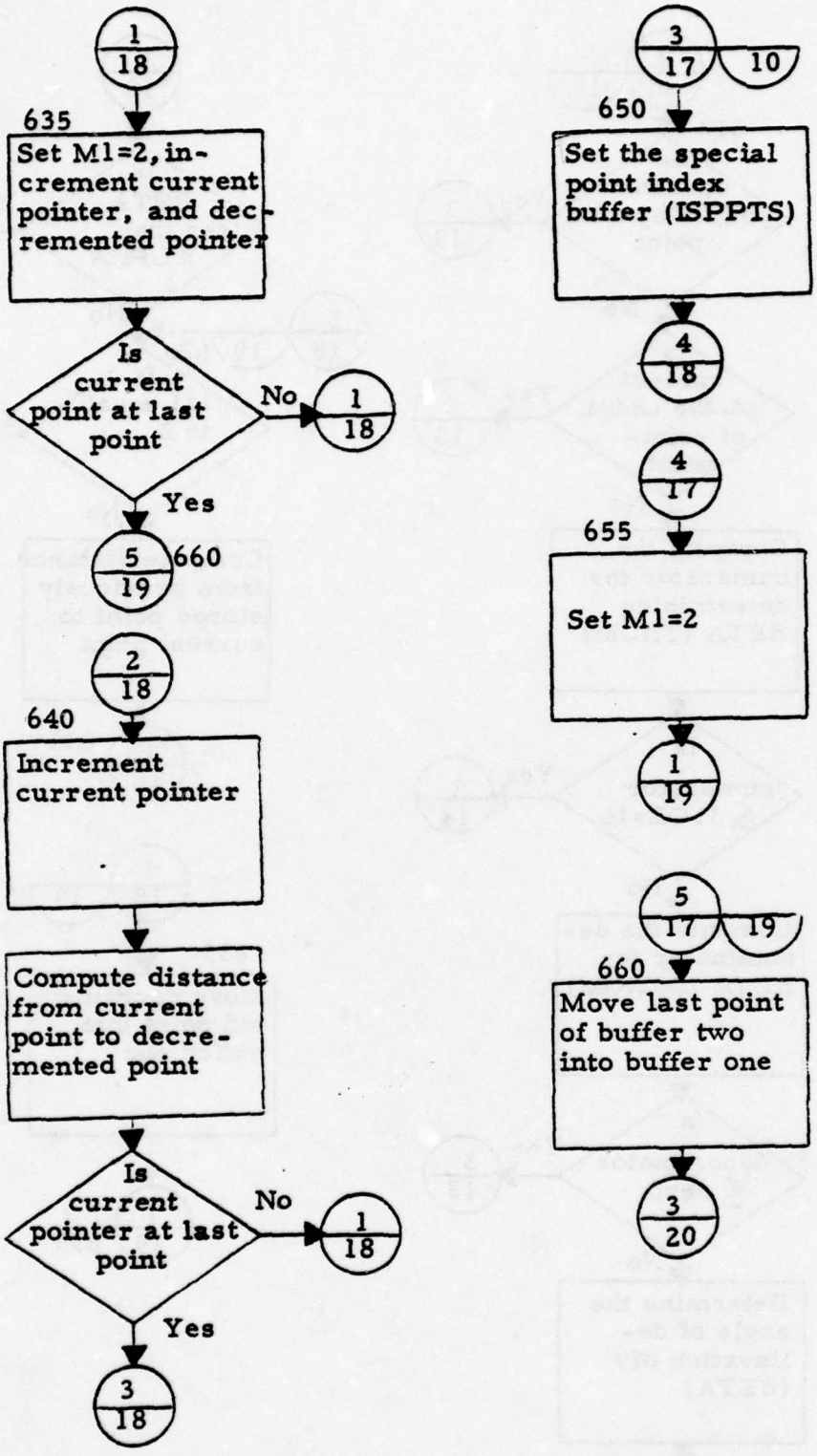



Figure III-18 SMOOTH Process Flow (Page 19 of 20)

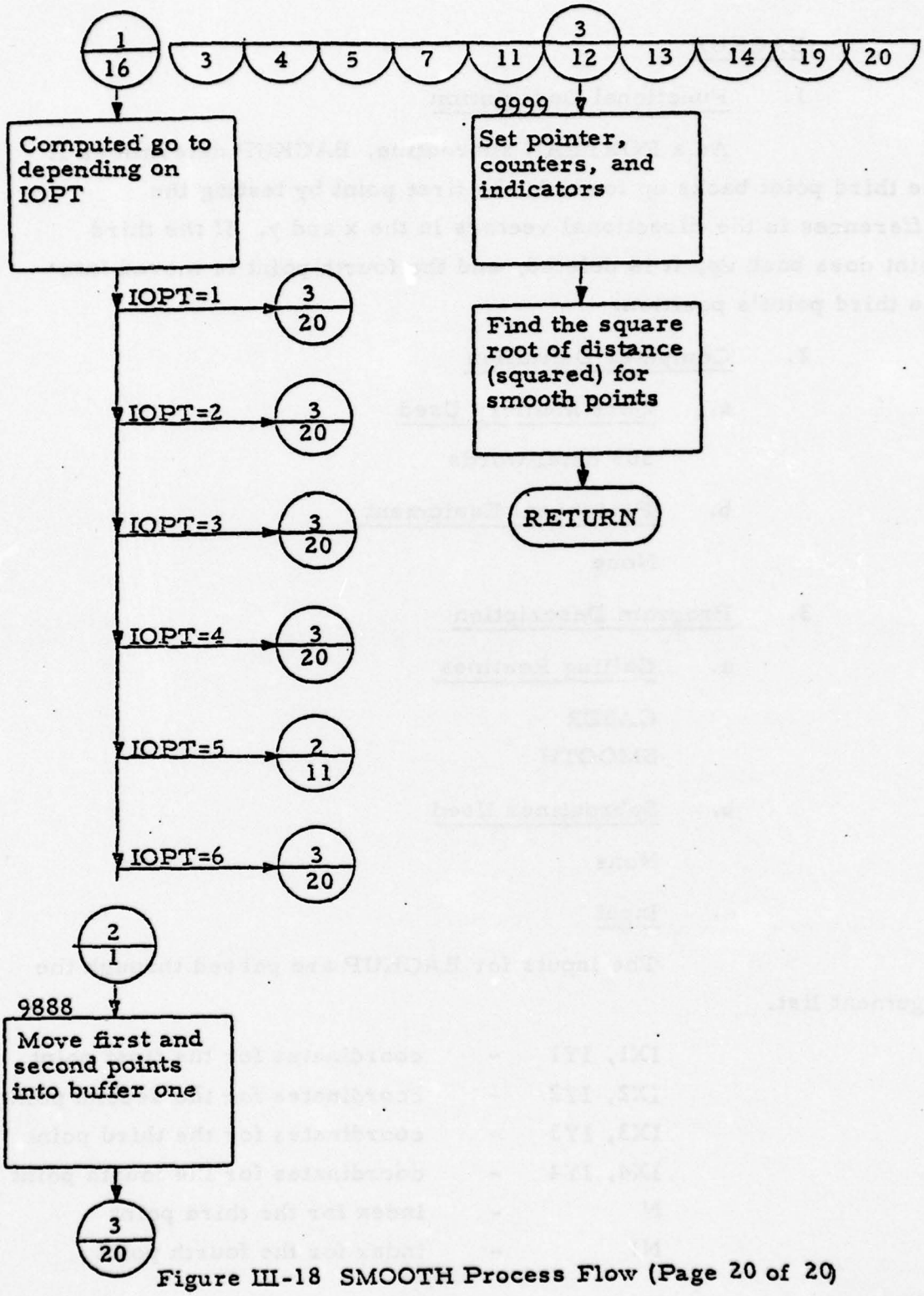



Figure III-18 SMOOTH Process Flow (Page 20 of 20)

## R. BACKUP

### 1. Functional Description

As a FORTRAN subroutine, BACKUP determines if the third point backs up towards the first point by testing the differences in the directional vectors in the x and y. If the third point does back up, it is deleted, and the fourth point is moved into the third point's position.

### 2. Computer Definition

#### a. Core Memory Used

303 octal words

#### b. Peripheral Equipment

None

### 3. Program Description

#### a. Calling Routines

CASER
SMOOTH

#### b. Subroutines Used

None

#### c. Input

The inputs for BACKUP are passed through the argument list.

| | | |
|---|---|---|
| IX1, IY1 | - | coordinates for the first point |
| IX2, IY2 | - | coordinates for the second point |
| IX3, IY3 | - | coordinates for the third point |
| IX4, IY4 | - | coordinates for the fourth point |
| N | - | index for the third point |
| N1 | - | index for the fourth point |

d. Output

The outputs for BACKUP are passed through the argument list.

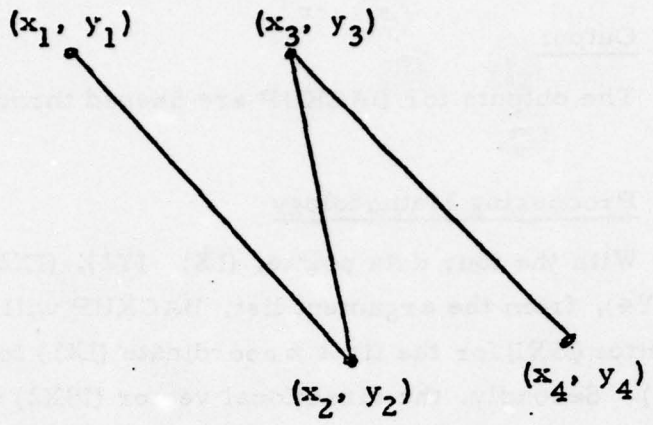
e. Processing Methodology

With the four data points, (IX1, IY1), (IX2, IY2), (IX3, IY3), and (IX4, IY4), from the argument list, BACKUP will first determine the directional vector (ISX1) for the first x coordinate (IX1) to the second x coordinate (IX2). Secondly, the directional vector (ISX2) will be determined from the second x coordinate to the third x coordinate (IX3). A test is then made to determine if ISX1 equals ISX2. If the test determines equality, the directional vectors ISY1 and ISY2 are determined as described above, but with the y coordinates. The equality of ISY1 and ISY2 will result in a return to the calling routine. With nonequality of ISY1 and ISY2, or with the nonequality of ISX1 and ISX2, BACKUP will proceed in the following fashion. A test is performed to determine if the ISY3 (the directional vector of the third point y coordinate to the fourth point y coordinate) is equal to ISY2. Nonequality of ISY3 and ISY2 results in the removal of the third point. The third point is removed by moving the fourth point's coordinate into the third point's coordinate location and setting the fourth index equal to the third's index. Equality of ISY3 and ISY2 results in another test with the directional vector ISX3 (x coordinate direction from second point to thrid point) and ISX2. With equality of ISX2 and ISX3, BACKUP returns control to the calling routine, and with nonequality, BACKUP removes the third point by replacing it with the fourth point as explained above. Control is then returned to the calling routine. Refer to Figure III-19 for the process flow diagram of BACKUP.

f. Calling Sequence

Call BACKUP (IX1, IY1, IX2, IY2, IX3, IY3, IX4, IY4, N, N1).

g. Major Algorithms




Given $\overline{X}_{ij}$ as the directional vector from the x coordinate of the i th point to the x coordinate of the j th point, such that:

$$\overline{X}_{ij} = \begin{cases} -1 & \text{if } x_j - x_i < 0 \\ 0 & \text{if } x_j - x_i = 0 \\ +1 & \text{if } x_j - x_i > 0 \end{cases}$$

Given $\overline{Y}_{ij}$ as the directional vector from the y coordinate of the i th point to the y coordinate of the j th point, such that:

$$\overline{Y}_{ij} = \begin{cases} -1 & \text{if } y_j - y_i < 0 \\ 0 & \text{if } y_j - y_i = 0 \\ +1 & \text{if } y_j - y_i > 0 \end{cases}$$

If $\overline{X}_{12} \neq \overline{X}_{23}$ or $\overline{Y}_{12} \neq \overline{Y}_{23}$

and

$\overline{Y}_{34} \neq \overline{Y}_{23}$ with $\overline{Y}_{23} \neq 0$,

then a backup condition occurs and $(x_3, y_3)$ is deleted.

If $\overline{X}_{12} \neq \overline{X}_{23}$ or $\overline{Y}_{12} \neq \overline{Y}_{23}$

and

$\overline{X}_{23} \neq \overline{X}_{34}$ with $\overline{X}_{23} \neq 0$,

then a backup condition occurs and $(x_3, y_3)$ is deleted.

4. Program Constants and Variables

| | | |
|---|---|---|
| N | - | index of the third point |
| N1 | - | index of the fourth point |
| ISX1 | - | directional vector in the x direction from the first to second point |
| ISX2 | - | directional vector in the x direction from the second to third point |
| ISX3 | - | directional vector in the x direction from the third to fourth point |
| ISY1 | - | directional vector in the y direction from the first to second point |
| ISY2 | - | directional vector in the y direction from the second to third point |
| ISY3 | - | directional vector in the y direction from the third to fourth point |
| IX1, IY1 | - | coordinates of the first point |
| IX2, IY2 | - | coordinates of the second point |
| IX3, IY3 | - | coordinates of the third point |
| IX4, IY4 | - | coordinates of the fourth point |

5. Error Conditions

None

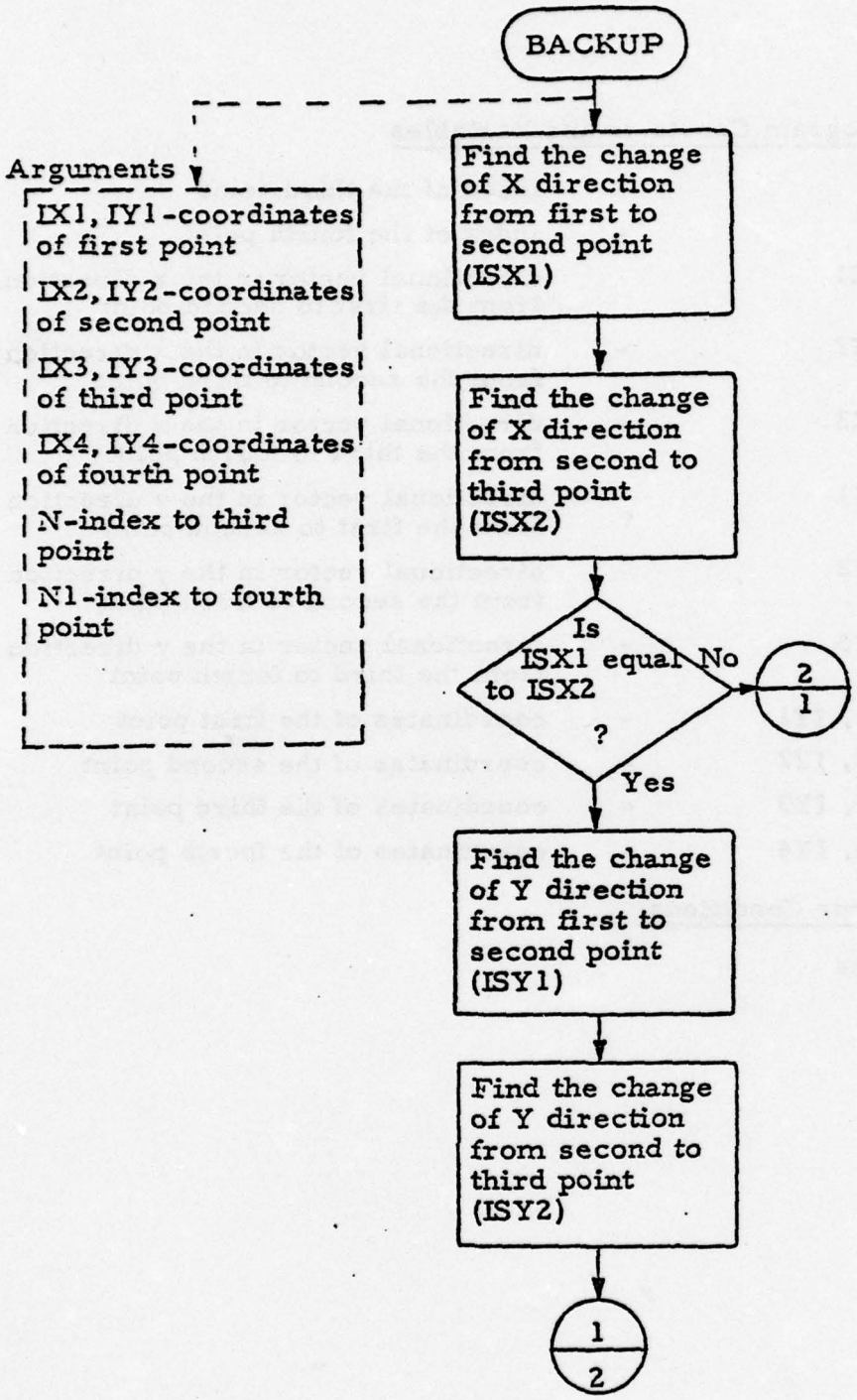



Figure III-19 BACKUP Process Flow (Page 1 of 4)

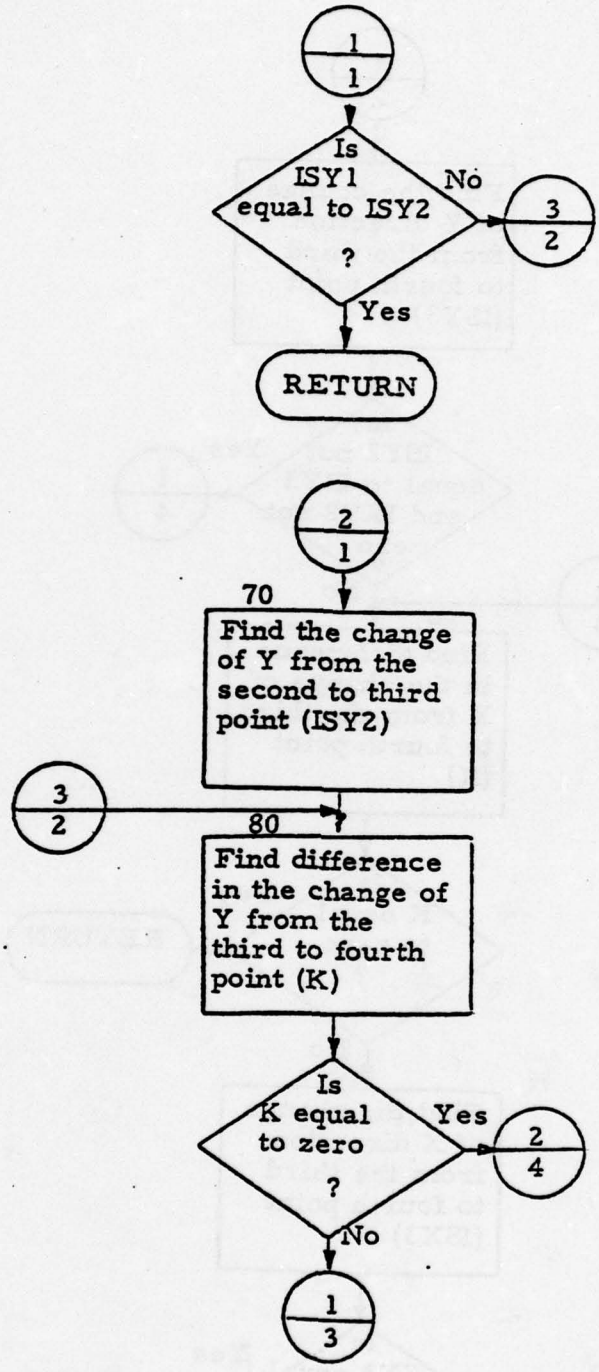



Figure III-19 BACKUP Process Flow (Page 2 of 4)

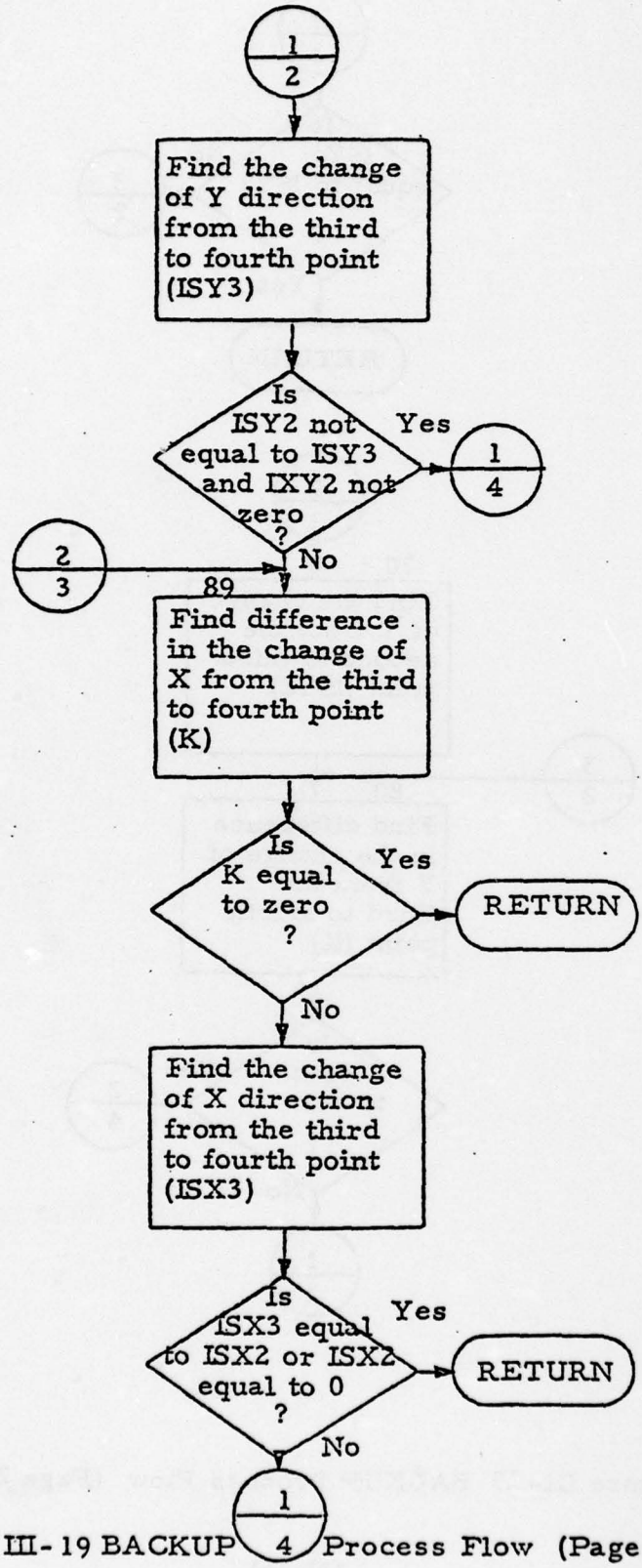



Figure III-19 BACKUP Process Flow (Page 3 of 4)

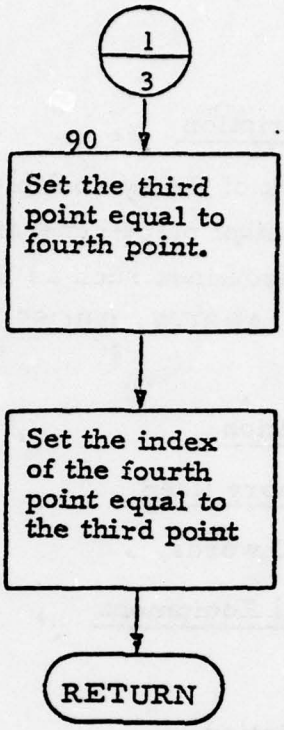



Figure III-19 BACKUP Process Flow (Page 4 of 4)

## S. SIMBOL

### 1. Functional Description

The primary task of the symbol piece controller subroutine SIMBOL is to control the symbol piece generation via interaction with symbol piece generation subroutines such as DASHER, SPACE, DOTTER, CIRCLE, TICKER, CASER, ARROW, CROSS, SQUARE, TRNGLE, PYRMID, and ARCORD.

### 2. Computer Definition

#### a. Core Memory Used

1667 octal words.

#### b. Peripheral Equipment

None

### 3. Program Description

#### a. Calling Routines

MONITOR.

#### b. Subroutines Used

| | | |
|---|---|---|
| DASHER | TICKER | SQUARE |
| SPACE | CASER | TRNGLE |
| DOTTER | ARROW | PYRMID |
| CIRCLE | CROSS | ARCORD |

#### c. Input

Primary input consists of data found in the GLSS blank common area (symbol specification directives) and (status indicator flags and pointers). The process control information is located in blank common area (symbol specification directives), FORTRAN mnemonic name ISYTP (symbol piece type) whose values are depicted in Table III-5.

| Symbol Piece Type | Numeric Value |
|---|---|
| LINE | 1 |
| DASH | 2 |
| SPACE | 3 |
| DOT | 4 |
| CIRCLE | 5 |
| TICK | 6 |
| HALF TICK | 7 |
| ALTERNATING HALF TICK | 8 |
| CASE | 9 |
| ARROW | 10 |
| HALF ARROW | 11 |
| CROSS | 12 |
| SQUARE | 13 |
| TRIANGLE | 14 |
| PYRAMID | 15 |
| ARC and CORD | 16 |

SYMBOL PIECE TYPE AND ASSOCIATED VALUES

TABLE III-5

d. Output

Output consists mainly of setting or resetting of various flags and pointers located in blank common area (symbol specification directives area). The primary output, mnemonic name ICURDX (current index pointer), controls the retrieval and storage of symbol piece generation.

e. Processing Methodology

Figure III-20 depicts the process flow of subroutine SIMBOL. Upon entry, the header input count (mnemonic IHEDIN) is checked, and if found to be different from the previously stored header count (mnemonic ITPHED), all SIMBOL subroutine flags and pointers are reset. The symbol piece types are then checked for certain combinations with the appropriate flags being set or reset. These combinations are: line, space, and tick (mnemonic ILSTK being set); dash, space, and tick (mnemonic IDSTK being set); dash, space, and case (menmonic ICSFLG being set); arrow (mnemonic IARFLG being set) when an arrow type is found. If the input header count and the previously stored header count are the same, subroutine SIMBOL call back flag is set to zero (ICLLBK). If the symbol piece call back flag is set (NUMCBK), control is passed to the appropriate subroutine. If the flag is not set, the symbol piece index pointer (mnemonic ISYDEX) is incremented by one and checked against the number of symbol pieces (NUMPEC). If ISYDEX is found to be greater than NUMPEC, ISYDEX is set to one or two, depending on the combination of flags set above. The symbol piece type numeric value is then used for transfer to the appropriate subroutine via a FORTRAN computed GO TO statement. If the symbol piece type numeric value is one (line), the symbol ready for output flag (mnemonic ISYRDY) is set. If there are more symbol pieces, the SIMBOL subroutine call back flag is set; otherwise it is cleared. Control is then returned to the calling routine MONITOR. If the symbol piece type numeric value is two (dash), process control is passed to subroutine DASHER

with the appropriate current index buffer pointer value set (mnemonic ICURDX). When the symbol piece type value is three, subroutine SPACE is called to generate a space on the data found via the current index buffer pointer. Upon returning from SPACE, control is passed to the above mentioned incrementing of the symbol piece index (ISYDEX). If the symbol piece type value is four (dot), subroutine DOTTER is called to generate a single coordinate point. When the symbol piece type is five (circle), symbol generation subroutine CIRCLE is executed. If the symbol piece type value is a six (tick), seven (half tick), or eight (alternating half tick), subroutine TICKER is called with the proper current index pointer value. When nine (case) is found in the symbol piece type, symbol generator subroutine CASER is performed again with the appropriate current index buffer value. If the symbol piece type value is ten (arrow) or eleven (half arrow), symbol piece generation subroutine ARROW is called. If the symbol piece type is twelve, subroutine CROSS is performed generating a cross symbol. If the symbol piece type is thirteen (SQUARE), symbol generator subroutine SQUARE is executed. If the symbol piece type is fourteen (TRIANGLE), symbol generator subroutine (TRNGLE) is performed. When the symbol type is fifteen (PYRMID) subroutine PYRMID is called to generate a pyramid at the coordinate value input. If the symbol piece type value is sixteen (ARC and CORD) subroutine ARCORD is called to generate an arc cord symbol. Upon returning from the symbol generation subroutines, exception SPACE stated, subroutine SIMBOL's call back flag is set; and if the number of symbol piece call back (mnemonic NUMCBK) is set, control is returned to the calling routine MONITOR. If the tally run out flag for buffer number one (ITELRN (1) is not set, control is again returned to MONITOR; if it is set, and the feature continuation flag is zero, subroutine SIMBOL's call back flag is set to zero with control being returned to MONITOR for the last time for the feature in question.

f. Calling Sequence

Call SIMBOL

g. Major Algorithms

None

4. Program Constants and Variables

Program variables interrogated by SIMBOL include:

| | | |
|---|---|---|
| IHEDIN | - | feature number input |
| ITPHED | - | temporary feature number |
| ILSTK | - | symbol types line, space, tick flag |
| IDSTK | - | symbol types dash, space, tick flag |
| ICSFLG | - | symbol types case flag |
| IARFLG | - | symbol types arrow flag |
| ICLLBK | - | subroutine SIMBOL call back flag |
| NUMCBK | - | number of symbol subroutine call backs |
| ISYDEX | - | symbol piece directive index pointer |
| NUMPEC | - | number of symbol pieces |
| ISYRDY | - | symbol ready for output flag |
| ICURDX | - | current buffer pointer (1-5) |
| ITELRN(1) | - | tally run out flag for buffer one |

5. Error Conditions

None

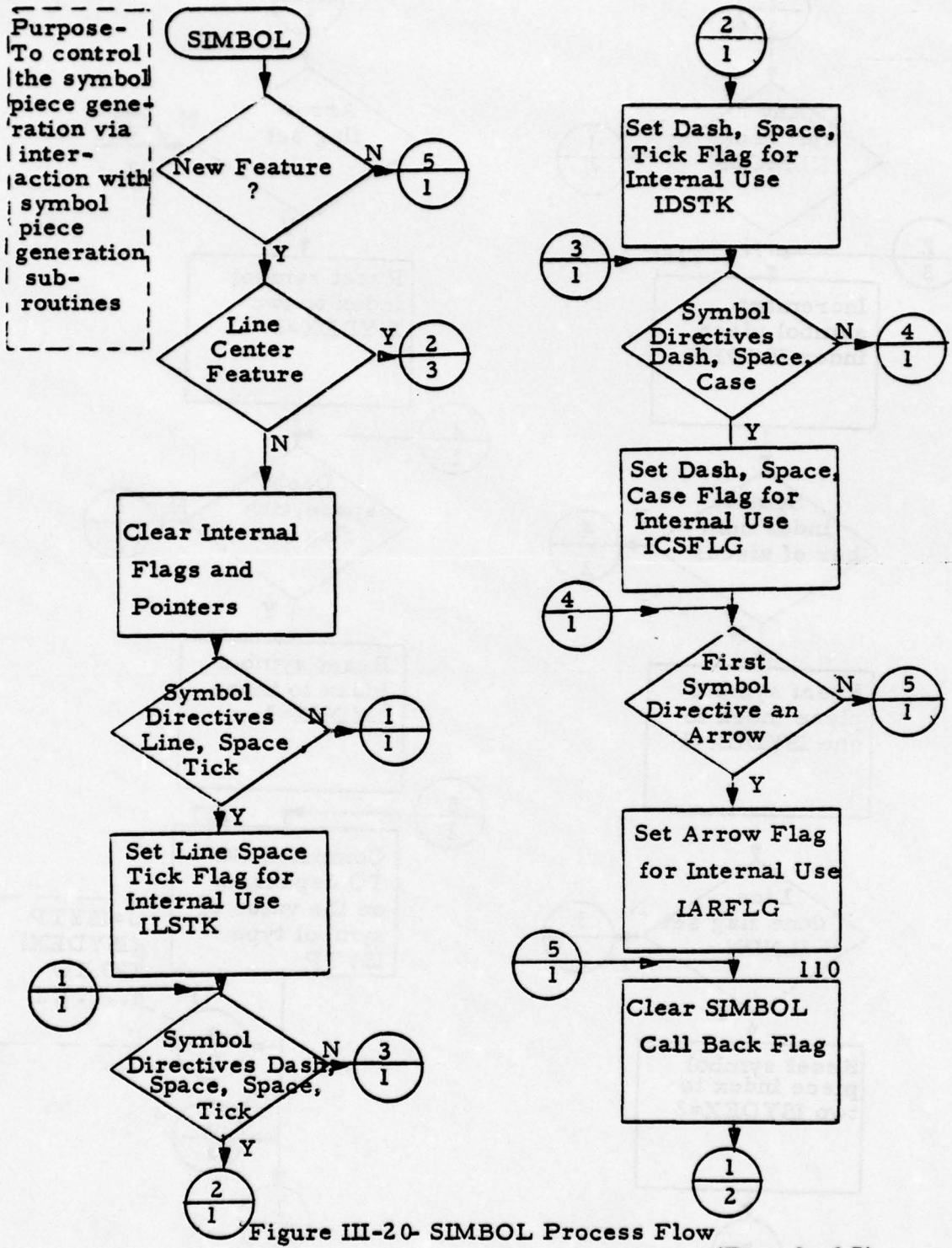



Figure III-20- SIMBOL Process Flow
(Page 1 of 7)

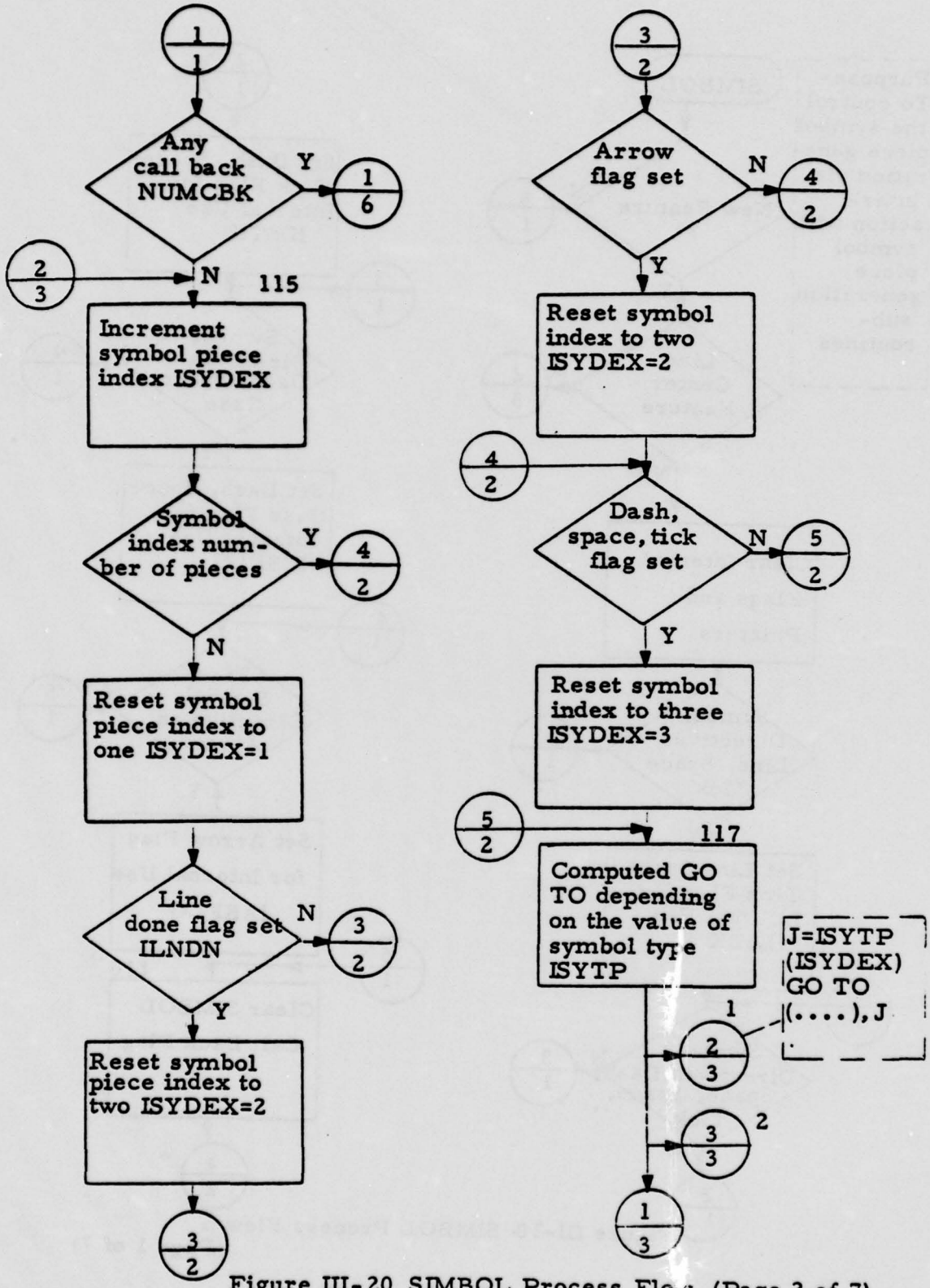



Figure III-20 SIMBOL Process Flow (Page 2 of 7)

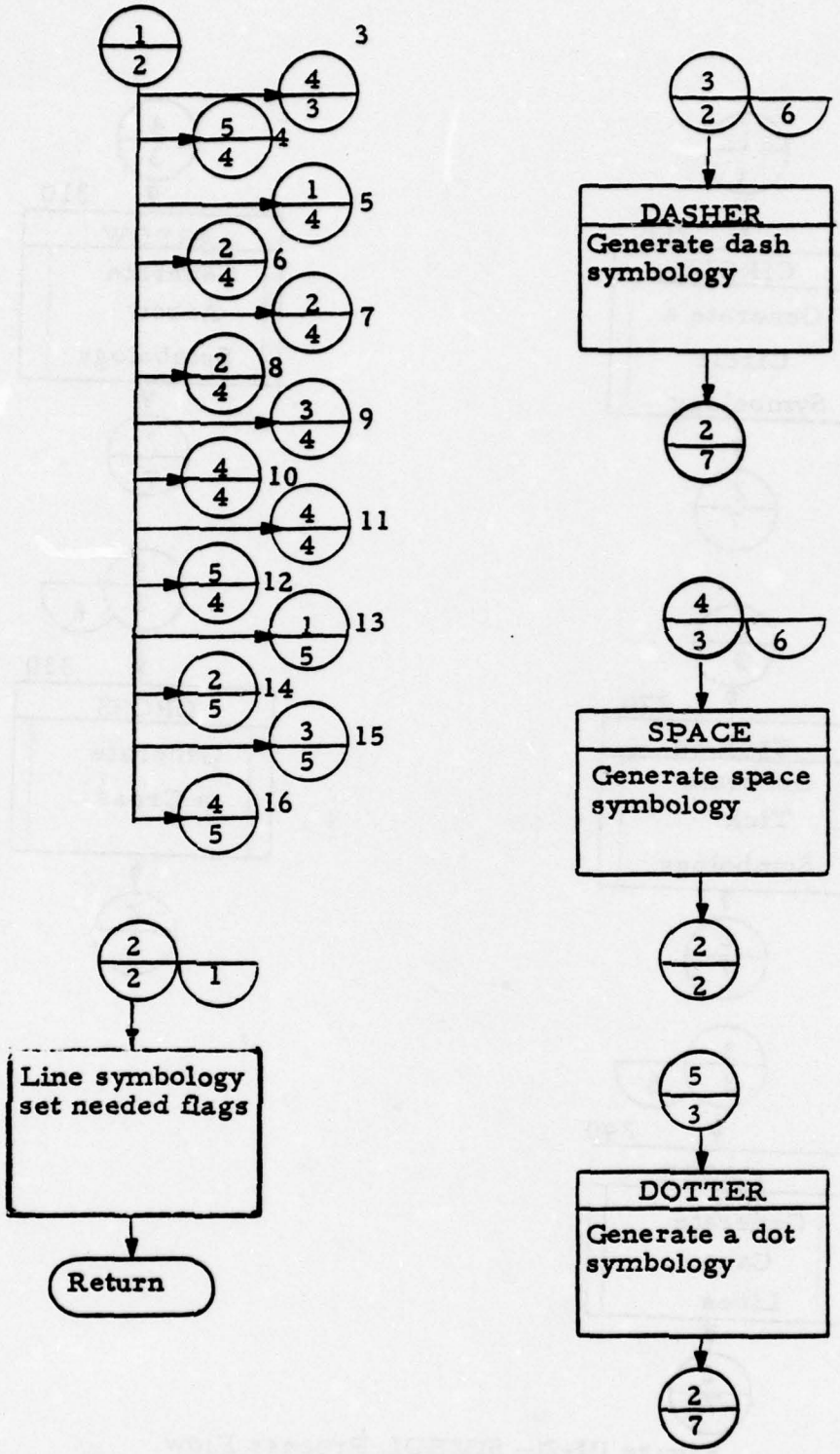



Figure III-20SIMBOL Process Flow (Page 3 of 7)

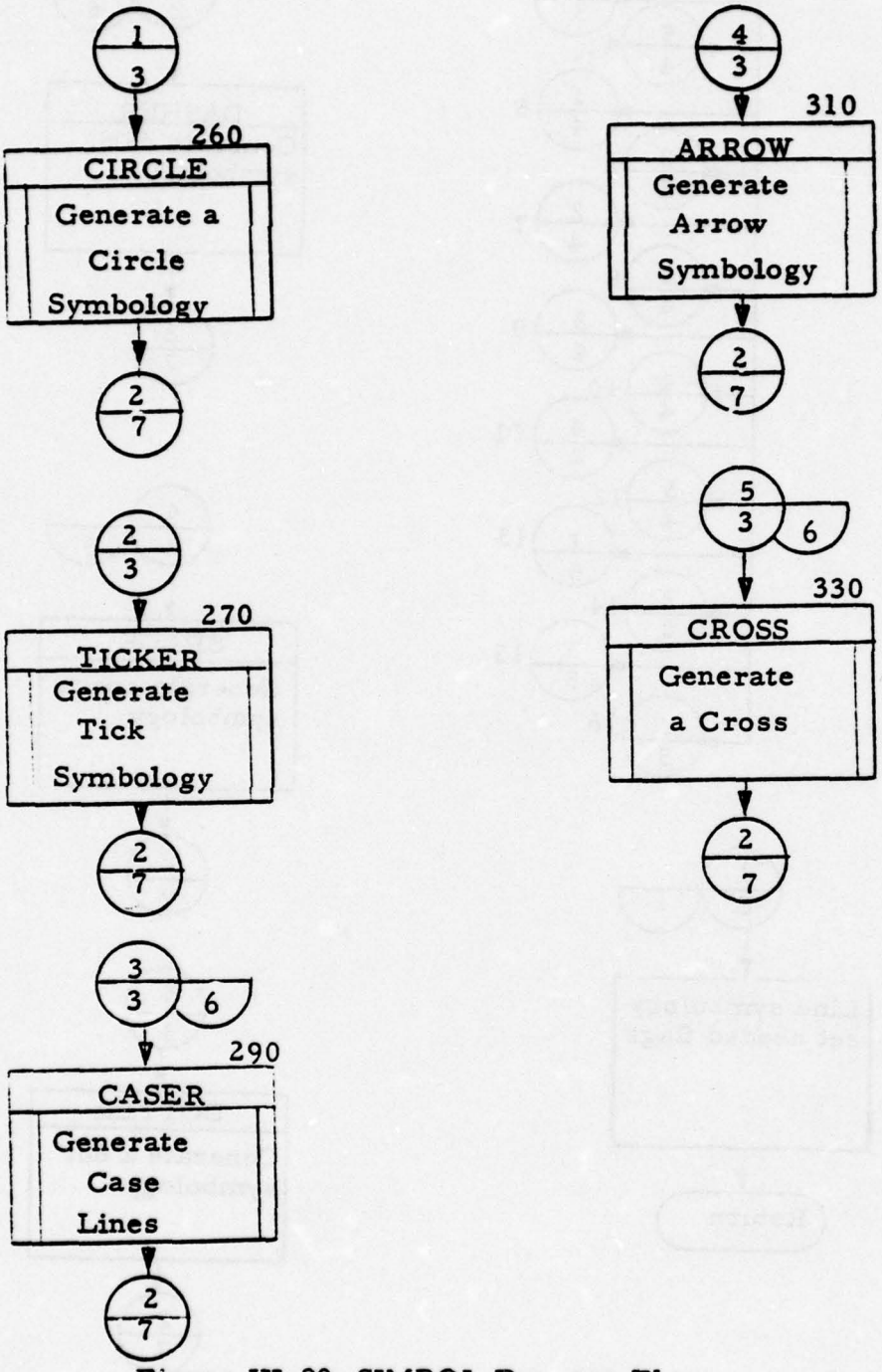



Figure III-20- SIMBOL Process Flow

(Page 4 of 7)

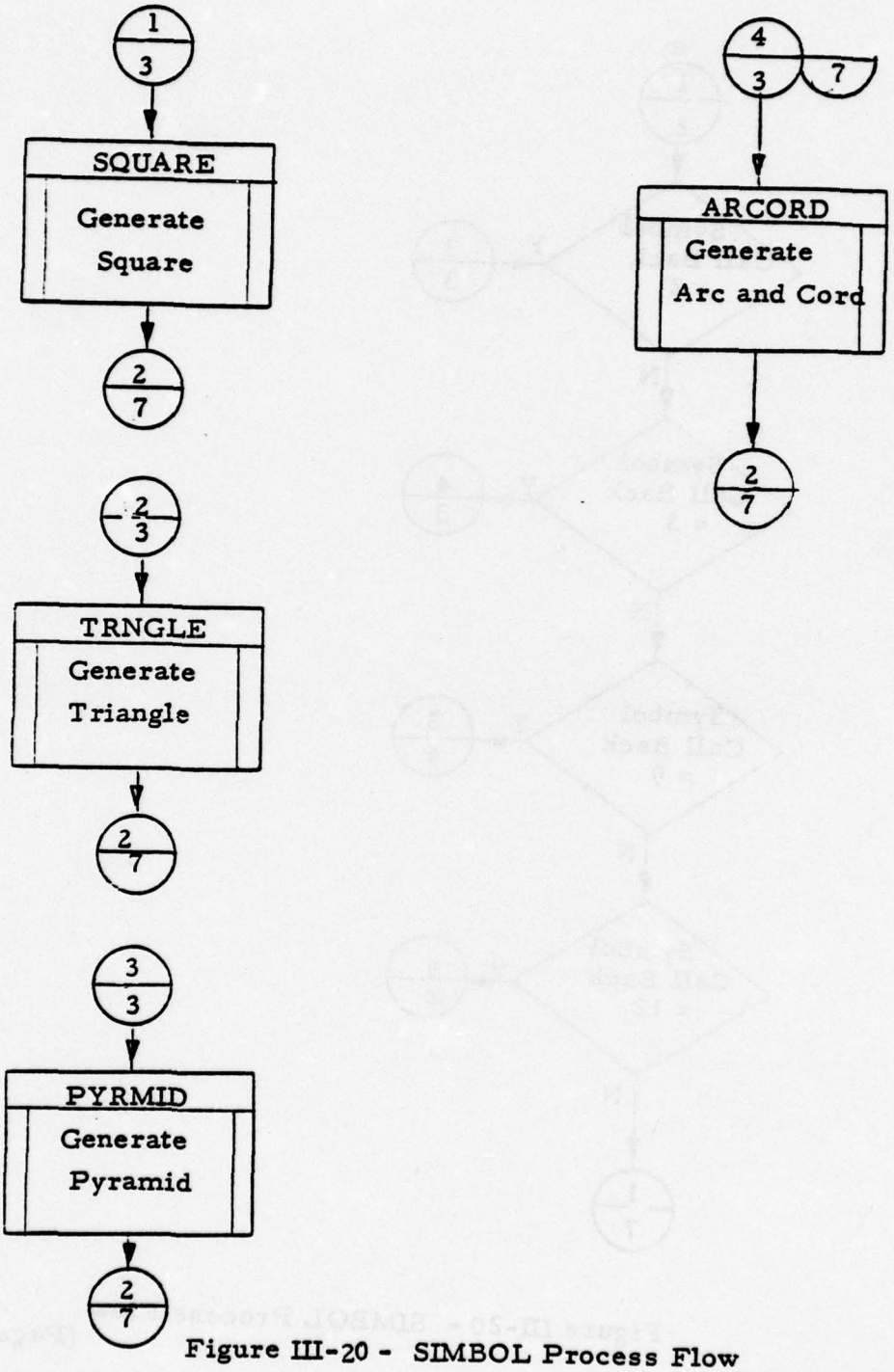



Figure III-20 - SIMBOL Process Flow

(Page 5 of 7)

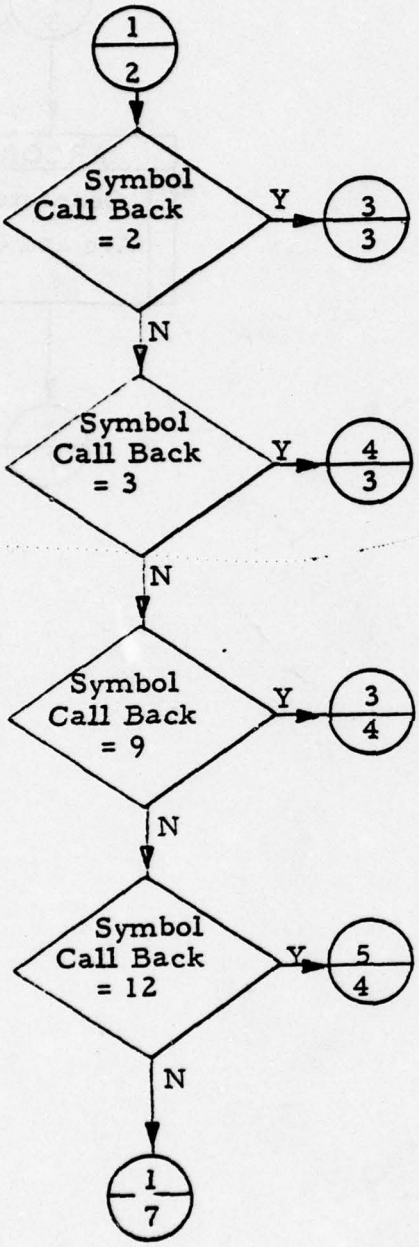



Figure III-20 - SIMBOL Process Flow (Page 6 of 7)

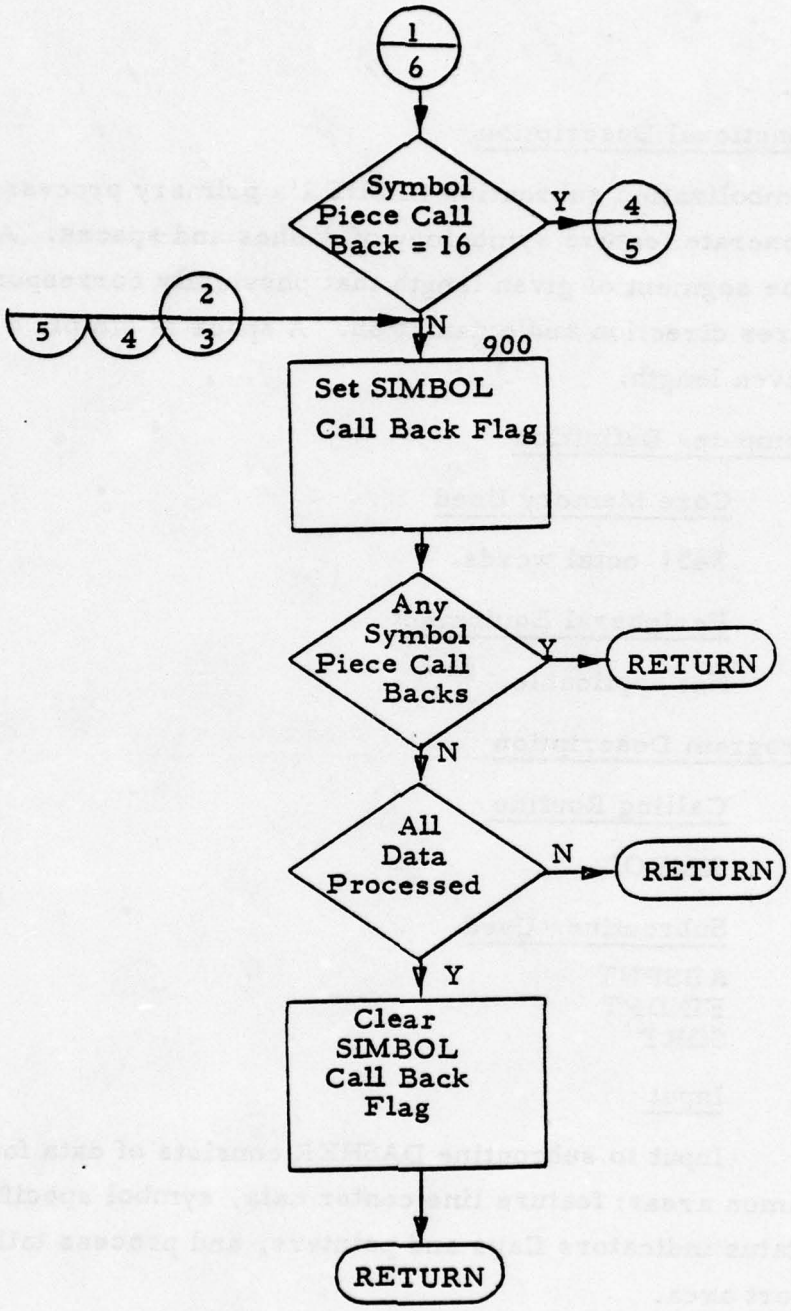



Figure III-20 - SIMBOL Process Flow

(Page 7 of 7)

## T. DASHER

### 1. Functional Description

Symbolization subroutine DASHER's primary processing tasks are to generate feature symbology of dashes and spaces. A dash symbol is a line segment of given length that physically corresponds to the input features direction and orientation. A space is simply a null dash for the given length.

### 2. Computer Definition

#### a. Core Memory Used

3451 octal words.

#### b. Peripheral Equipment

Not applicable.

### 3. Program Description

#### a. Calling Routine

SIMBOL

#### b. Subroutines Used

ABSPNT
FINDPT
SQRT

#### c. Input

Input to subroutine DASHER consists of data found in the blank common areas: feature line center data, symbol specification directives, status indicators flags and pointers, and process tally summary report area.

#### d. Output

Output will consist of a dash symbol coordinate and

distance data stored in mnemonic IXYZ output buffer. It will also consist of status indicator flags and pointers and tally summary report data.

e. Processing Methodology

Upon entry to subroutine DASHER, needed flag and pointers are cleared and/or set. Figure III-21 depicts the process flow of DASHER. The "Dasher" call back flag is interrogated and if "false" normal processing is continued. The "true" discussion of the call back flag will be described below. The input index pointer for buffer number one is saved (IPTDX (1)) with the symbol type (ISYTPE) being interrogated. If the symbol type is a space (ISYTPE = 3), the input and output pointers are saved for possible use later in the program (dashing at the end of a feature). The symbol size (ISYSZ (ISYDEX)) is moved into menmonic name ISIZE with the distance location IDIST being cleared. If the symbol type is a dash, the X, Y coordinate and associated distance are placed into the output buffer dictated by ICURDX. The number of dashed points (NDSHPT) and number of points (NPTS) used to generate the dash in question are then incremented. If the symbol type is a space and the number of special points is not equal to zero, subroutine ABSPNT is called to determine if the coordinate point in question is an absolute point. If the point is an absolute coordinate, the symbol piece index pointer ISYDEX is decremented by one, the input index pointer is reset to the start of the space and control is passed to the beginning of the processing. If the summed distance (IDIST) is less than the symbol size (ISIZE), the summed distance is incremented by the next distance in the input buffer. The input index pointer (IPTDX(1)) is then incremented by one and checked against the number of points for buffer one (NUMPTS(1)). If the index pointer is found to be greater than or equal to the number of points in buffer one and the

feature continuation flag is set (IFTCNT) the "dasher" call back flag is assigned and the output buffer index pointer saved (mnemonic ISVPTX). If the smooth option is two or greater, the coordinate values generated for the partial dash are saved (interval buffer ISXYZ) and process control returned to the calling routine. On the next entry to DASHER, the aforementioned call back flag is interrogated and when found true it is reset along with the output buffer pointer. If the smooth option is two or greater, the previously saved coordinate values are moved to blank common output buffer mnemonic IXYZ. If the index pointer is found to be greater than or equal to the number of points and the symbol type in question is a space, the input and output index pointers (IPTDX(1) and IPTDX (ICURDX)) are reset with the last of the feature being moved to the output buffer. The symbol ready for output flag is set along with the tally run out flag for buffer one (ITELRN(1)). The number of points in the dash are then moved to the output number of points location, mnemonic name NUMPTS, dictated by the current index pointer ICURDX. Process control is then returned to the calling routine SIMBOL. The above process is repeated until the summed distance (IDIST) is equal to or greater than the symbol piece size (ISIZE). Subroutine FINDPT is then called to calculate the intermediate x, y coordinate and distance generated by the summed distance overrun. This point and associated distances are then placed into the input and output buffers. If the symbol type (ISYTPE) is a dash, the symbol directive index (ISYDEX) is incremented along with the number of dashes generated (NUMDSH). Control is then passed to the start of the processing where the space symbol is generated. If the symbol type is a space the symbol ready for output flag (ISYRDY) is set with control being passed to the calling routine SIMBOL.

f. Calling Sequence

Call DASHER

g. Major Algorithms

None

4. Program Constants and Variables

Program variables interrogated by DASHER include:

ISYRDY - Symbol ready for output flag

ICURDX - Current buffer pointer (1-5)

NUMPTS (ICURDX) - No. of points in buffer ICURDX

ISYTP (ISYDEX) - Symbol types

IXYZ (N, N) - X, Y Coordinate and distance buffer

ICORDX (ICURDX) - Starting index pointer into one of the five IXYZ buffers

IPTDX (N) - Current index pointers

ITELRN (1) - Tally run out flags for buffer one

5. Error Conditions

None

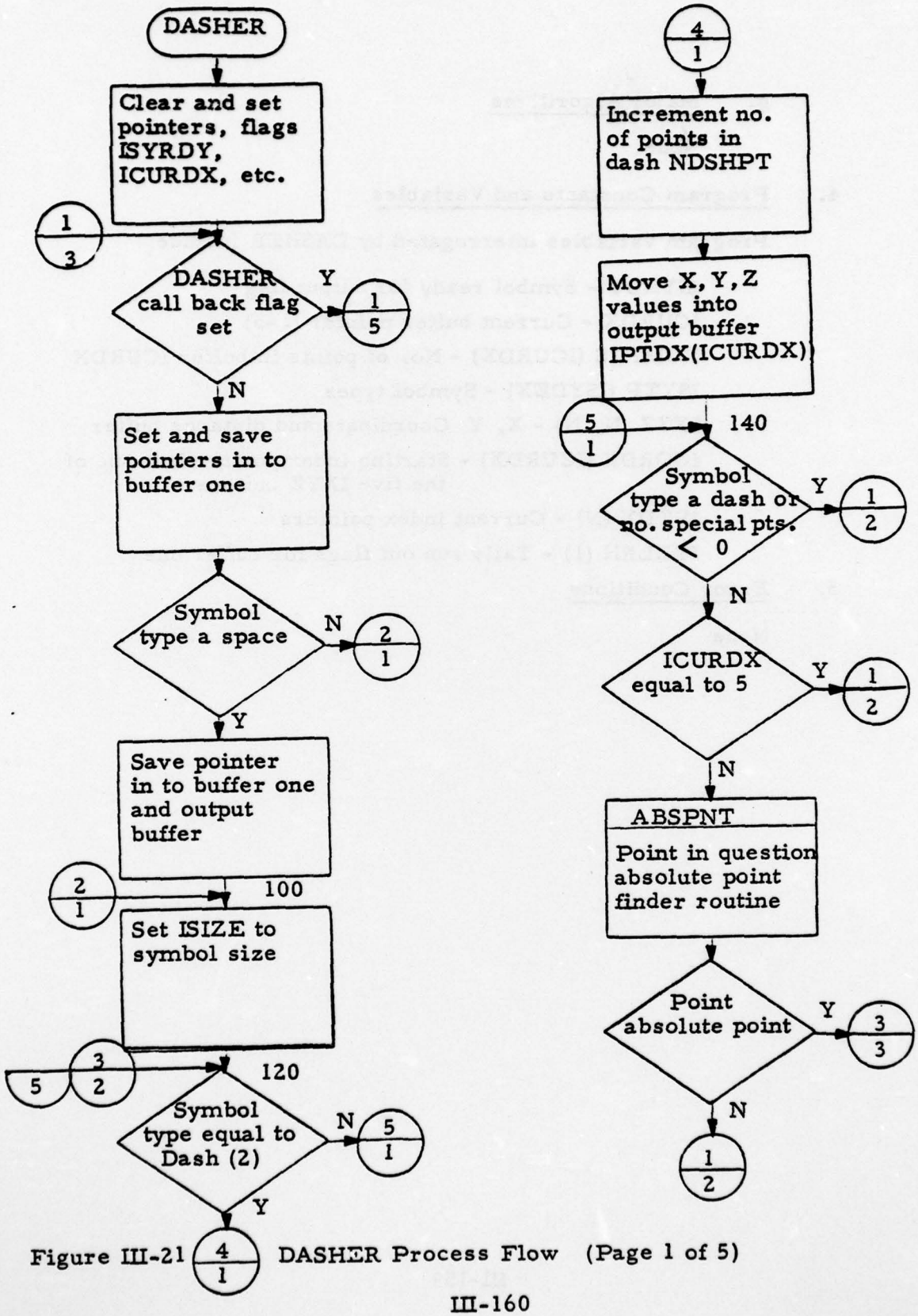



Figure III-21 DASHER Process Flow (Page 1 of 5)

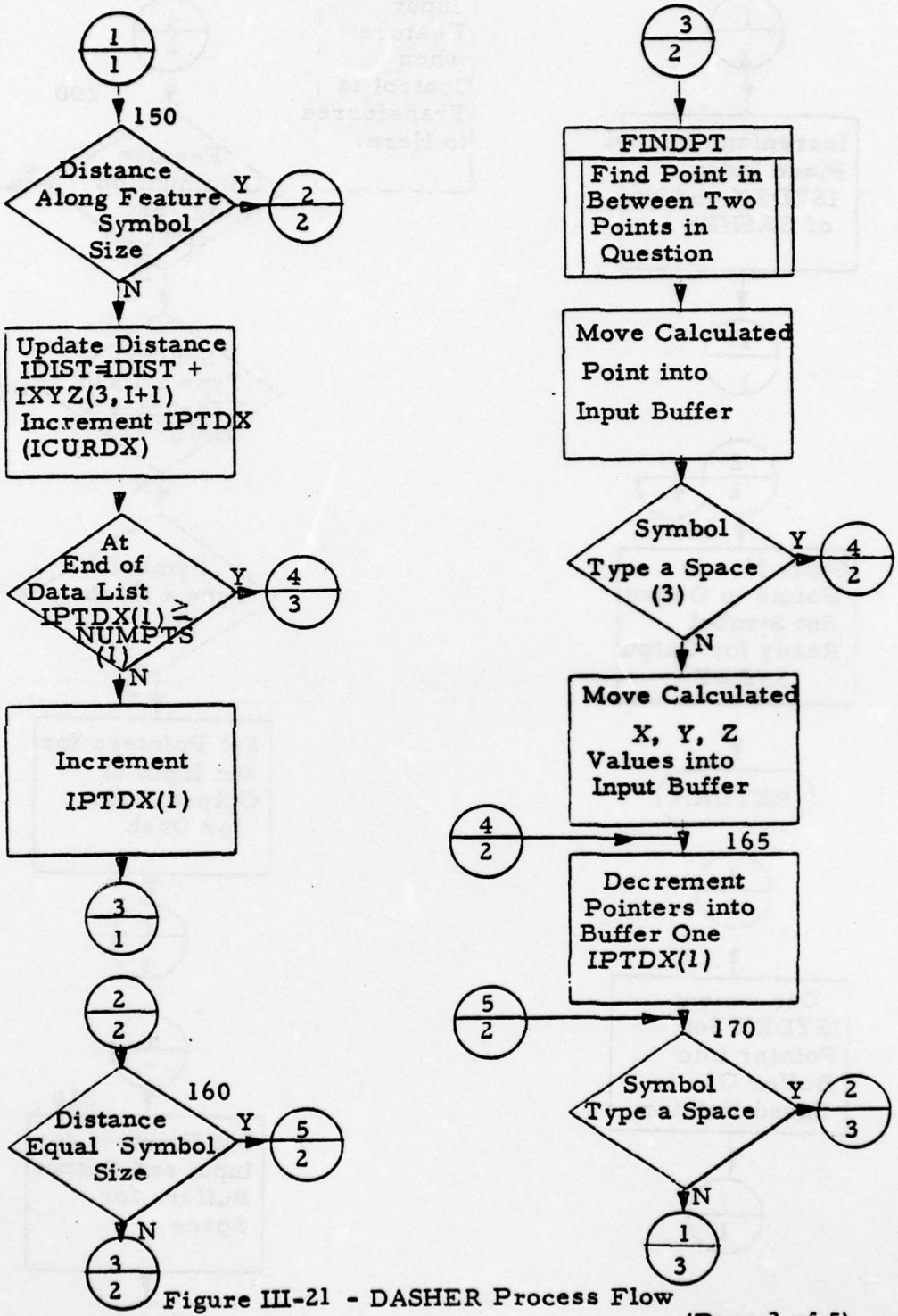



Figure III-21 - DASHER Process Flow

(Page 2 of 5)

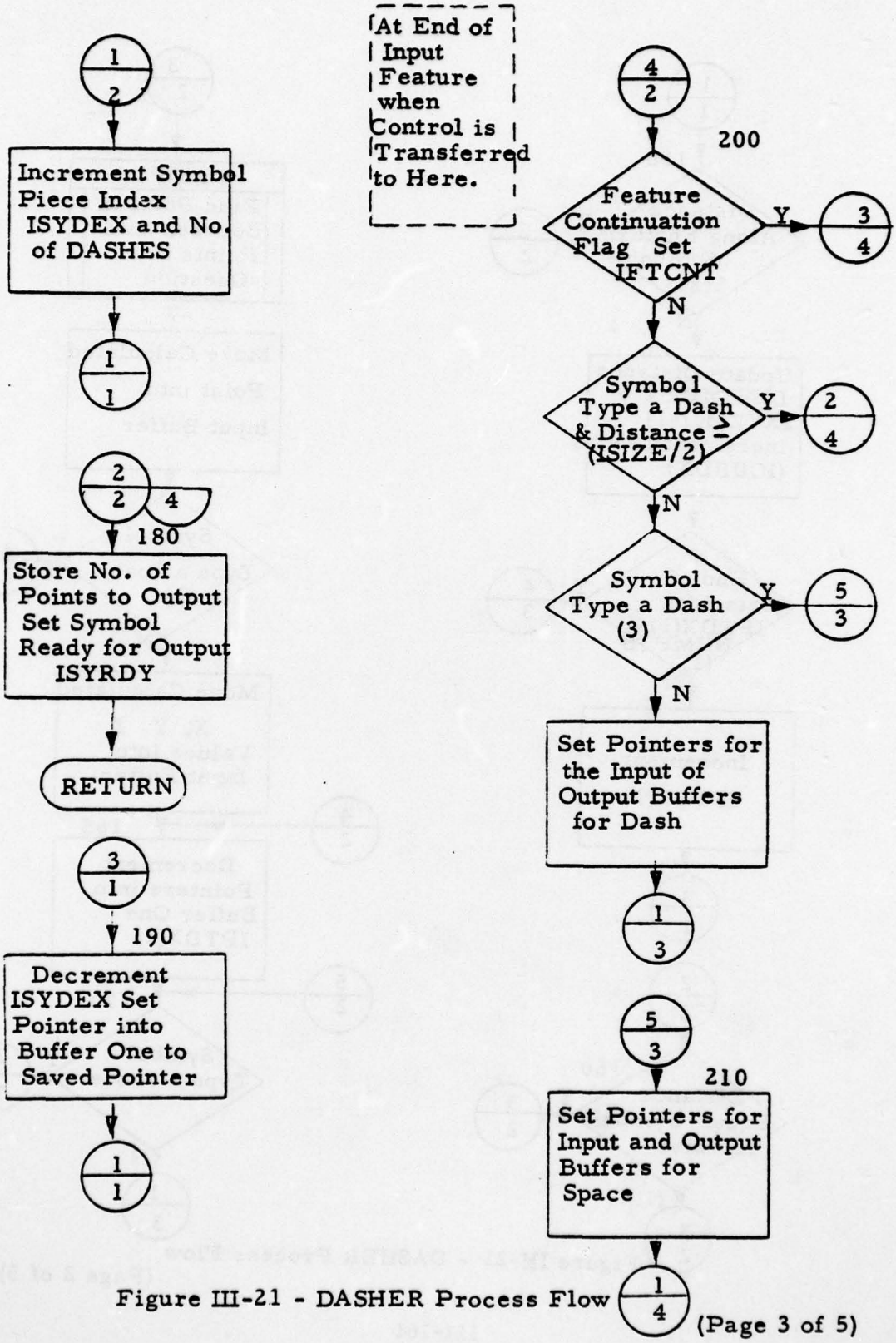



Figure III-21 - DASHER Process Flow (Page 3 of 5)

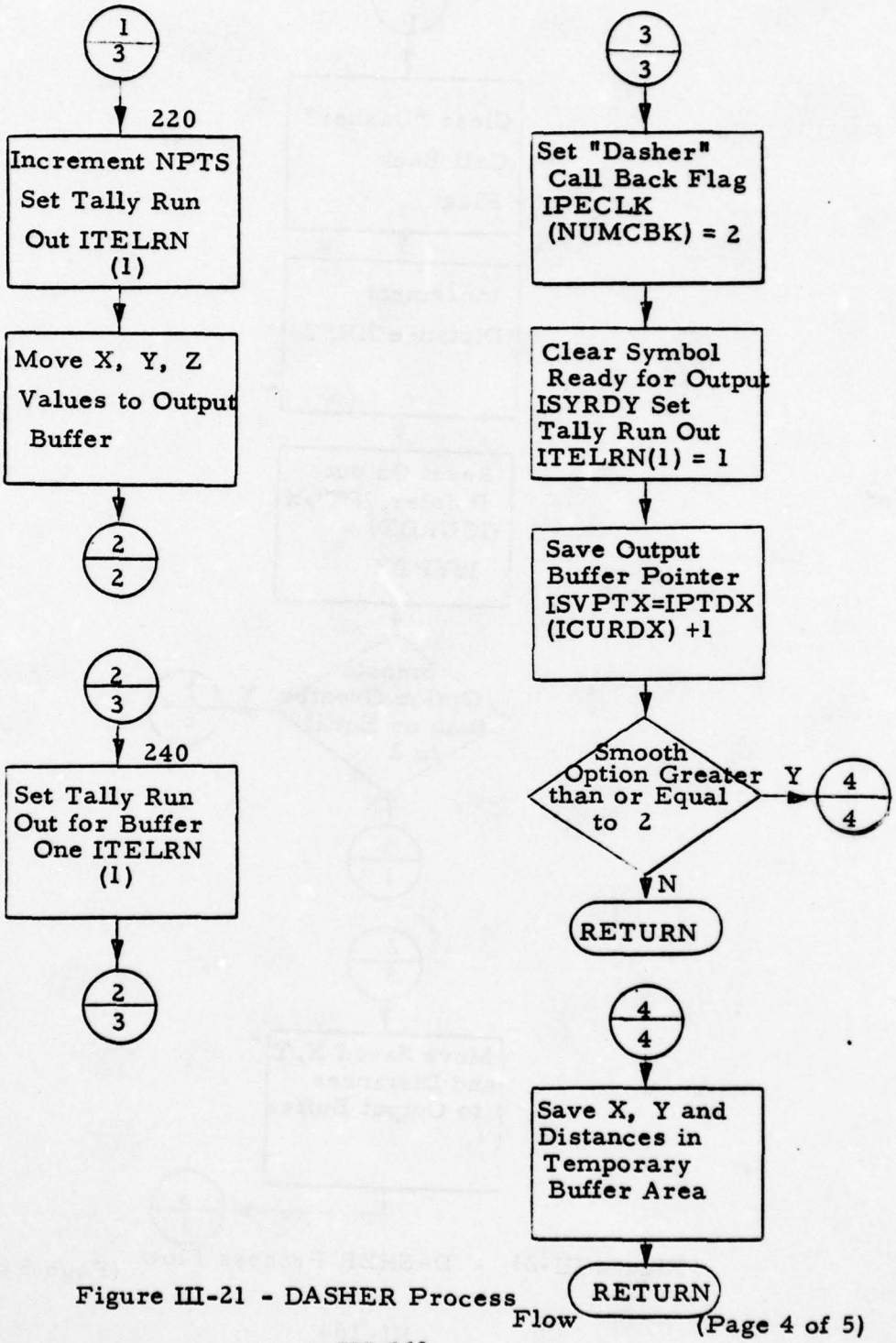



Figure III-21 - DASHER Process Flow (Page 4 of 5)

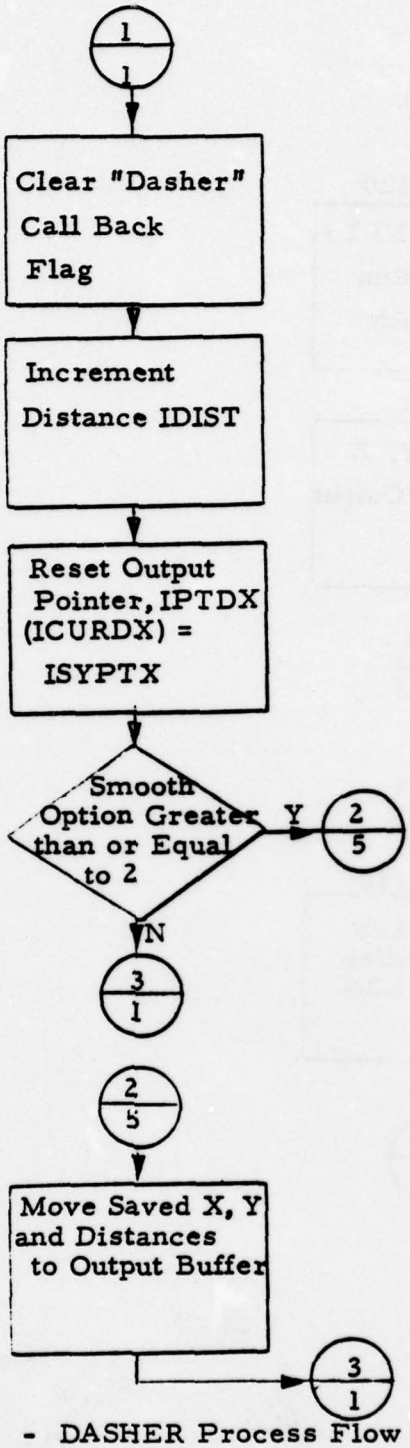



Figure III-21 - DASHER Process Flow (Page 5 of 5)

## U. Subroutine ABSPNT

### 1. Functional Description

Subroutine ABSPNT's major task is to interrogate and report absolute (special point) coordinate point determination. An absolute point is a coordinate point that cannot be moved or deleted so not to alter the characteristic of the feature or feature segment.

### 2. Computer Definition

#### a. Core Memory Used

55 octal words

#### b. Peripheral Equipment

Not applicable

### 3. Program Description

#### a. Calling Routines

DASHER
TICKER

#### b. Subroutines Used

None

#### c. Input

The input consists of data found in common area C2 (feature line center data), namely mnemonic ISPPTS, which contains index value of the absolute points within the IXYZ coordinate buffer and the present index pointer in question.

#### d. Output

The sole output is a flag (mnemonic IANS) which contains either zero (index pointer not an absolute point) or one (index pointer is at an absolute point).

e. Processing Methodology

Figure III-22 depicts the process flow of subroutine ABSPNT. On entry, mnemonic IANS (flag for absolute point) is cleared. The input index value, found in mnemonic J, is compared with the index values found in buffer ISPPTS. If a comparison is found, IANS is set to one. If a comparison is not found, process control is returned to the calling routine.

f. Calling Sequence

Call ABSPNT (J, IANS)

| | | |
|---|---|---|
| J | - | index pointer of coordinate in question |
| IANS | - | set to one when index pointer value (J) is found in special point buffer ISPPTS |
| IANS | - | set to zero when index pointer is not found in buffer ISPPTS |

g. Major Algorithms

Not applicable

4. Program Constants and Variables

| | | |
|---|---|---|
| ISPPTS | - | buffer containing special point index values found in IXYZ buffer |

5. Error Conditions

None

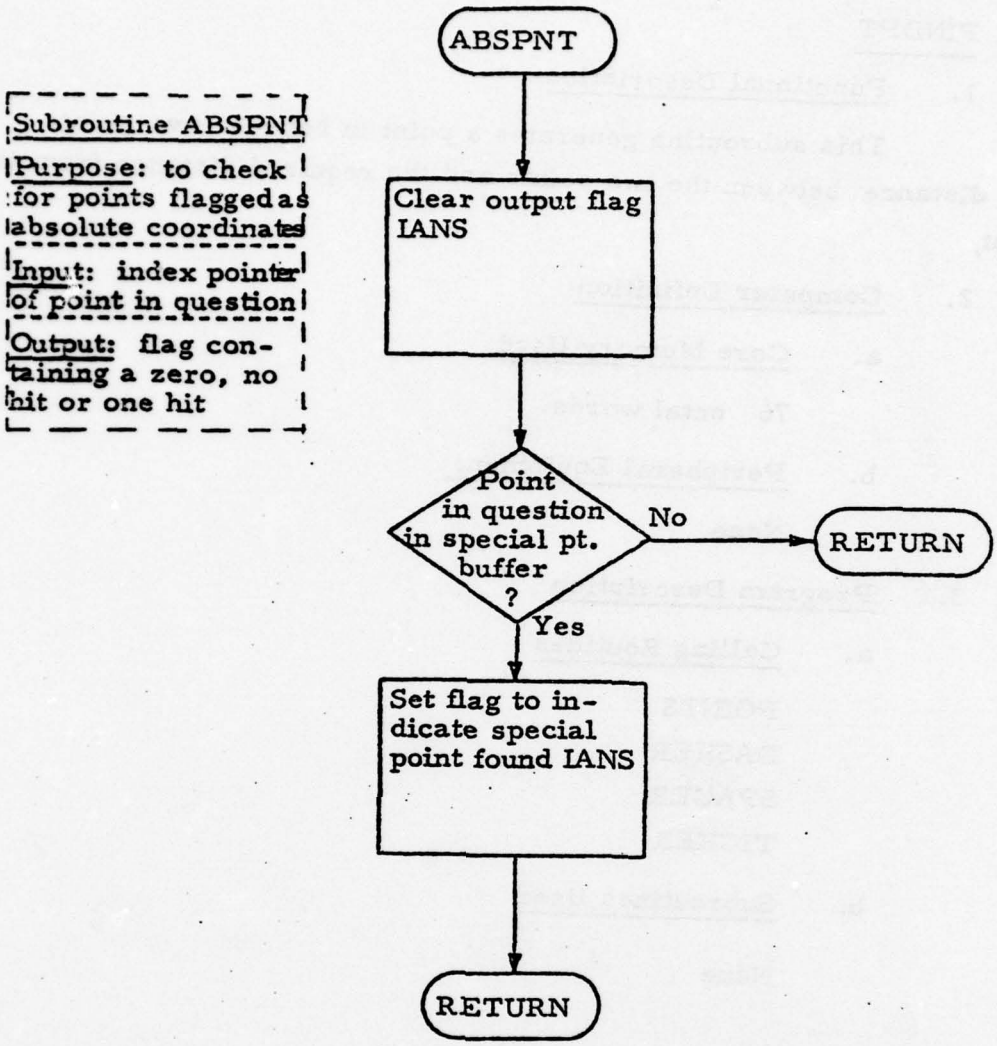



Figure III-22 ABSPNT Process Flow (Page 1 of 1)

## V. FINDPT

### 1. Functional Description

This subroutine generates a point in between two points given the distance between the two points and the required distance from the first point.

### 2. Computer Definition

#### a. Core Memory Used

76 octal words.

#### b. Peripheral Equipment

None

### 3. Program Description

#### a. Calling Routines

POINTS
DASHER
SPACER
TICKER

#### b. Subroutines Used

None

#### c. Input

| | | |
|---|---|---|
| IX1, IY2 | - | X, Y coordinates for first point. |
| IX2, IY2 | - | X, Y coordinates for second point. |
| ID | - | Distance from the first point to second point. |
| ID1 | - | Distance from the first point to generated point. |

#### d. Output

| | | |
|---|---|---|
| IX, IY | - | X, Y coordinates for generated point. |

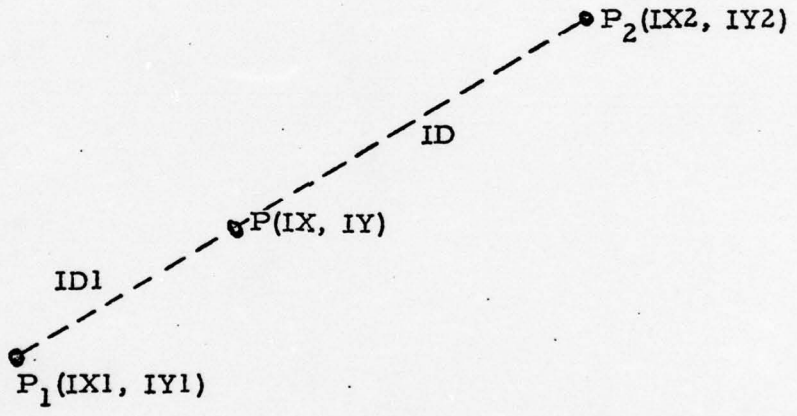
e. Processing Methodology

This subroutine, FINDPT, which is written in FORTRAN first computes the ratio (in floating point) of the distance from the first point to the generated point to the distance from the first point to the second point. The X, Y coordinates of the generated point are found as outlined in the following algorithm. See Figure III-23 for the processing flow diagram.

f. Calling Sequence

Call FINDPT (IX, IY, IX1, IY1, IX2, IY2, ID, ID1).

g. Major Algorithms




ID1 = distance squared from point $P_1$ to point P.
ID = distance squared from point $P_1$ to point $P_2$.
IX = IX1 + (IX2 - IX1) · ID1/ID
IY = IY1 + (IY2 - IY1) · ID1/ID

4. Program Constants and Variables

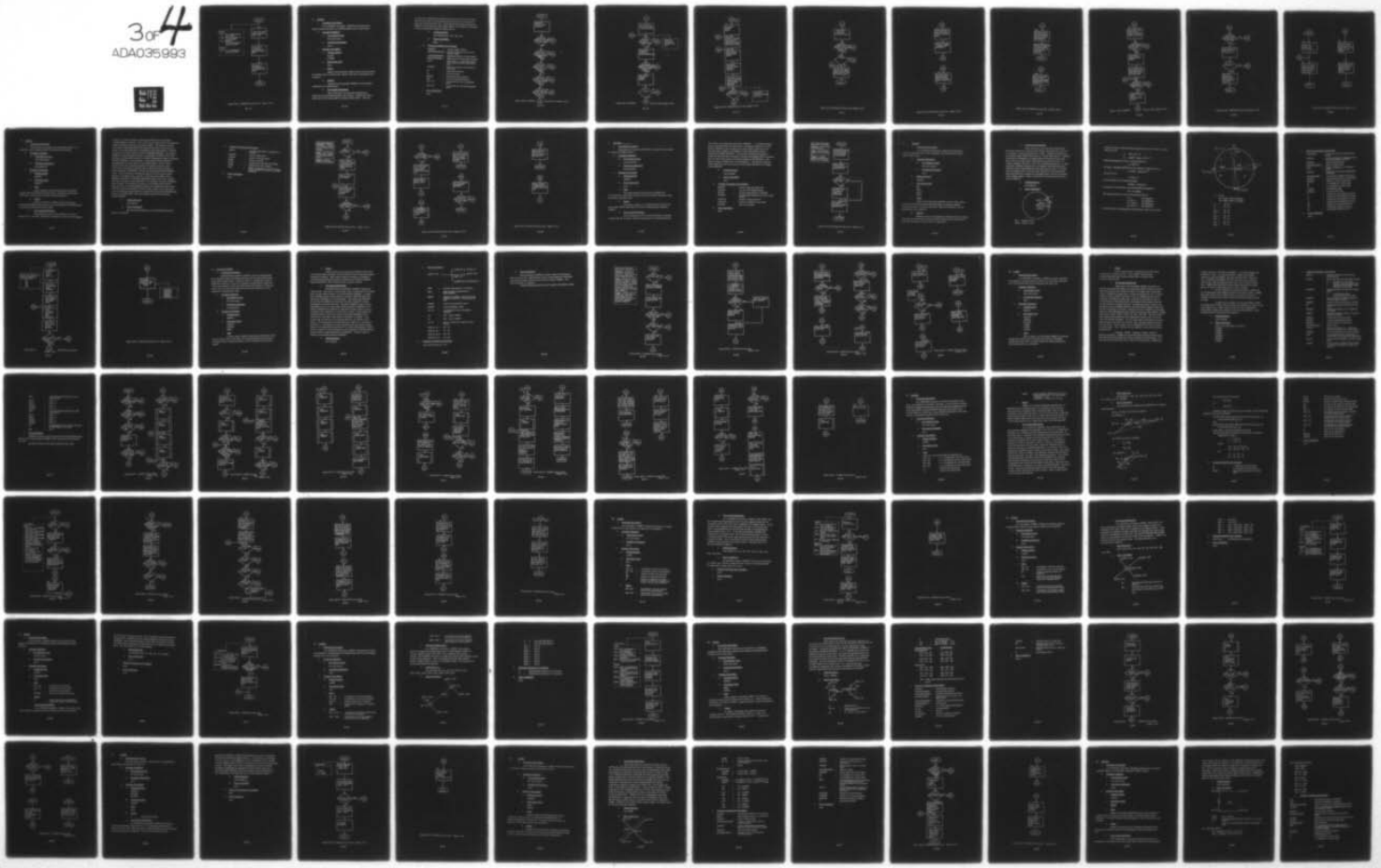
None

5. Error Conditions

None

AD-A035 993 PRC INFORMATION SCIENCES CO MCLEAN VA F/G 8/2
ACS SYMBOLIZATION FOR DMAAC. VOLUME II. COMPUTER PROGRAM DOCUME--ETC(U)
NOV 76 P D BELL, J A NEUFFER, M L TAYLOR F30602-75-C-0319
UNCLASSIFIED RADC-TR-76-334-VOL-2 NL

3 OF 4
ADA035993



F
5993

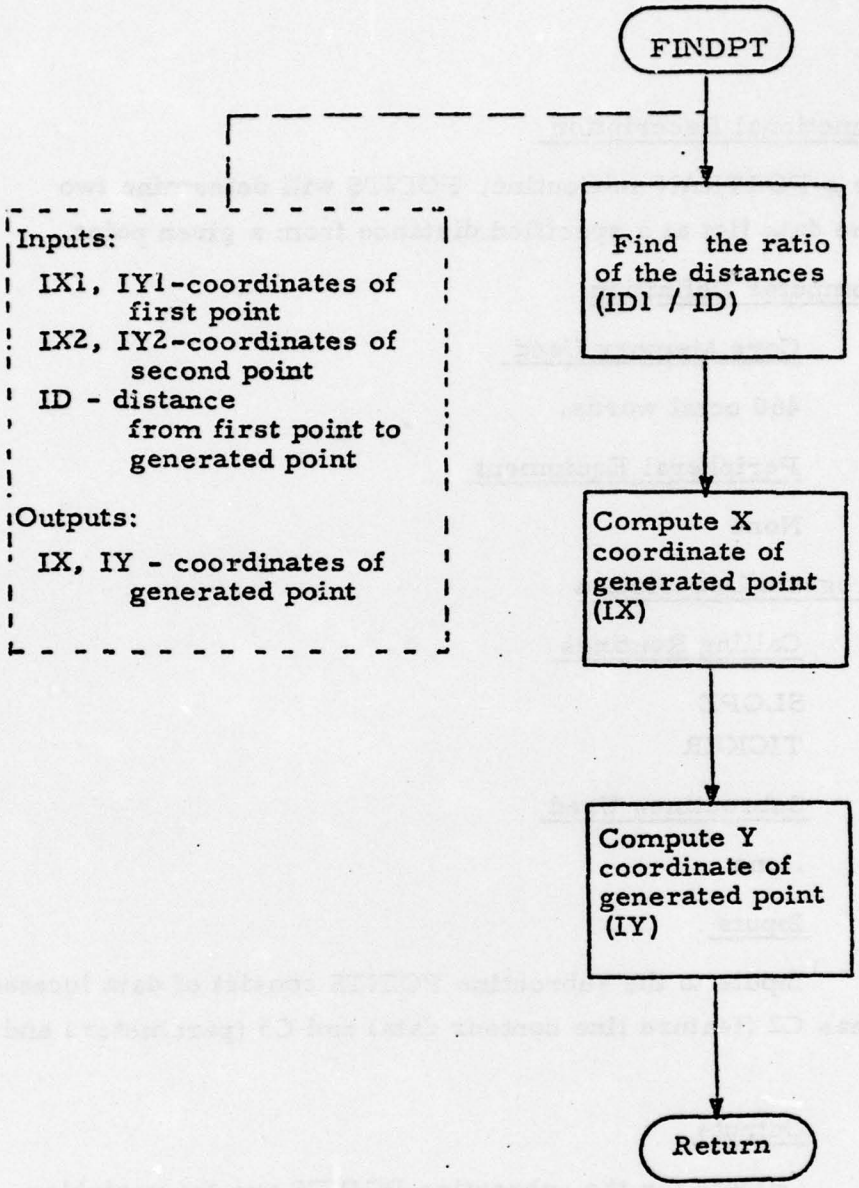



Figure III-23 FINDPT Process Flow (Page 1 of 1)

W. POINTS

1. Functional Description

As a FORTRAN subroutine, POINTS will determine two points along the data list at a specified distance from a given point.

2. Computer Definition

a. Core Memory Used

460 octal words.

b. Peripheral Equipment

None

3. Program Description

a. Calling Routines

SLOPE
TICKER

b. Subroutines Used

None

c. Inputs

Inputs to the subroutine POINTS consist of data located in common areas C2 (feature line contour data) and C5 (parameters and variables).

d. Outputs

Outputs for the subroutine POINTS are the variables contained in the argument list.

e. Processing Methodology

When called upon, the subroutine POINTS first initializes the appropriate pointers anc counters. POINTS then determines where the current point is located in IXYZ buffer. From the

current point, POINTS will interpret backwards in the data list to find the first point (IX1, IY1) and then interpret forward in the data list to find the second point (IX2, IY2). After finding both interpreted points, POINTS returns control to the calling routine. The method is described in the processing flow diagram, Figure III-24.

f. Calling Sequence

Call POINTS (IX1, IY1, IX2, IY2).

g. Major Algorithms

None

4. Program Constants and Variables

| | | |
|---|---|---|
| ICURDX | - | current buffer pointer |
| NUMPTS(ICURDX) | - | number of points in data list or current buffer |
| ICORDX(ICURDX) | - | beginning index for current buffer |
| IPTDX(ICURDX) | - | index to the current pointer pointer |
| IXYZ | - | buffer with x, y coordinate points with distance between successive pair |
| IMAXDT | - | maximum distance considered as trace data |
| IT | - | cumulated distances |
| M | - | last point index |
| ISPDST | - | input interpreted distance |
| IDH | - | half of interpreted distance |
| IX1, IY1 | - | coordinates for first interpreted point |
| IX2, IY2 | - | coordinates for second interpreted point |

5. Error Conditions

None

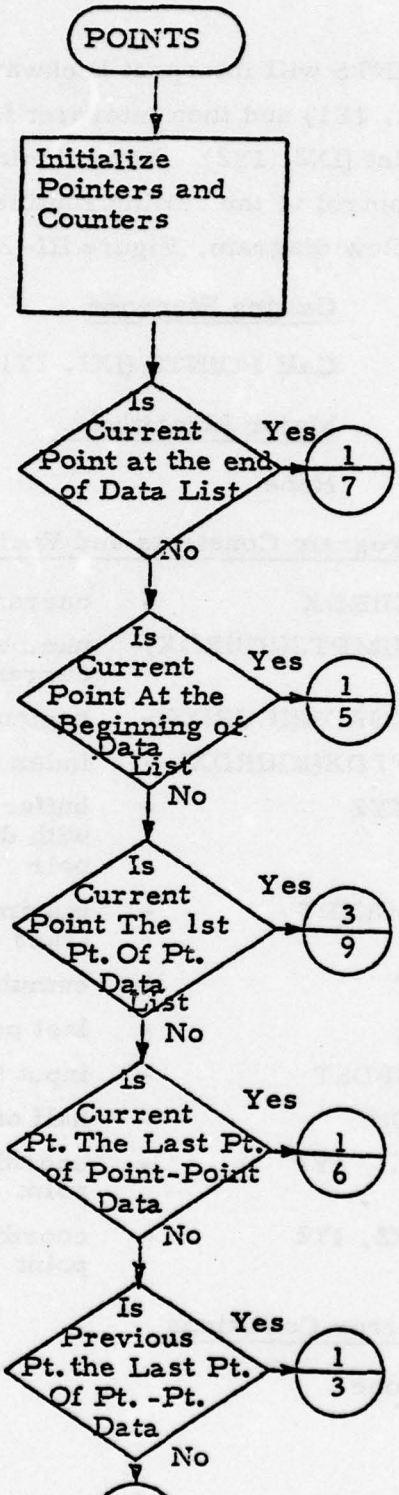



Figure III-24 POINTS Process Flow (Page 1 of 9)

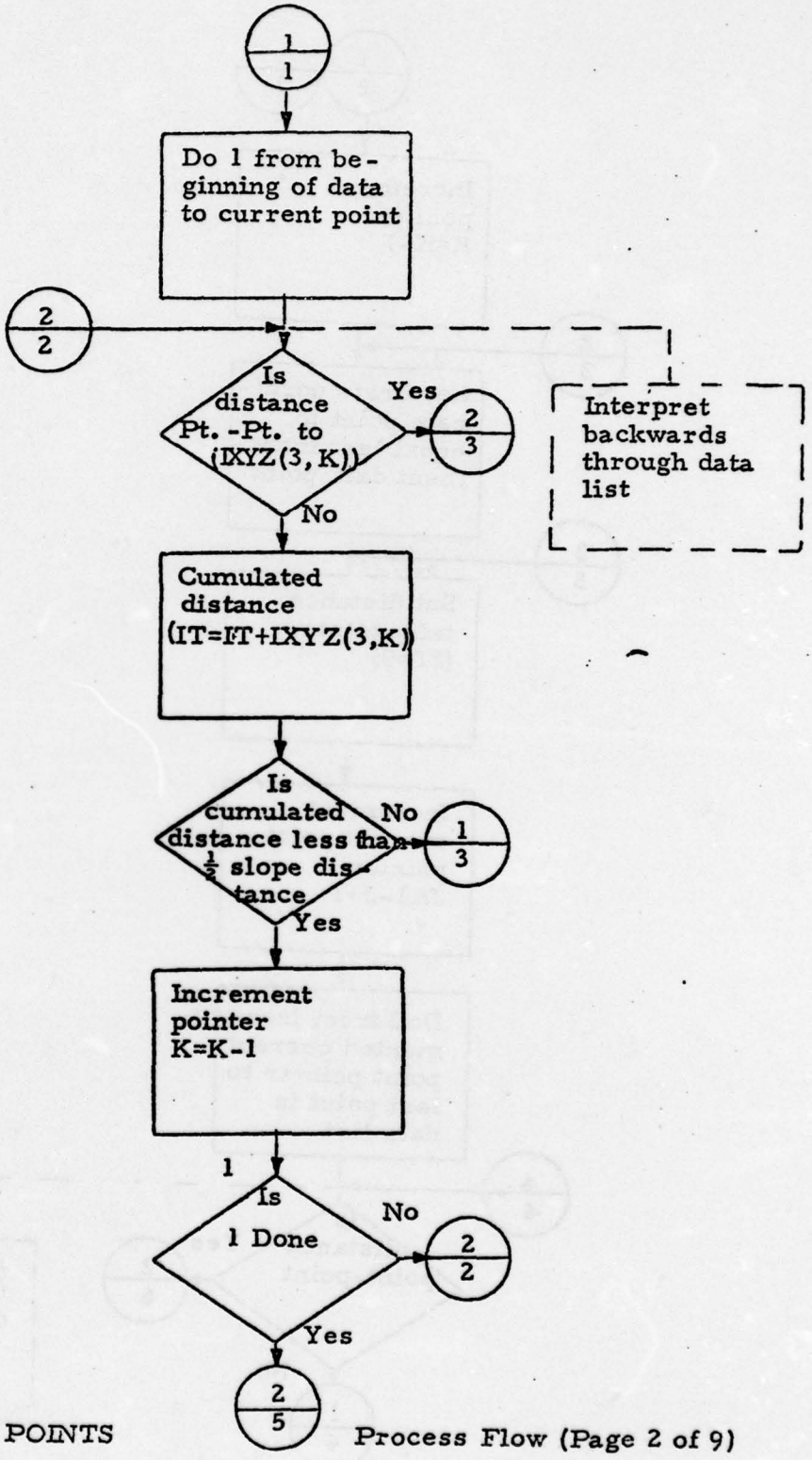



Figure III-24 POINTS Process Flow (Page 2 of 9)

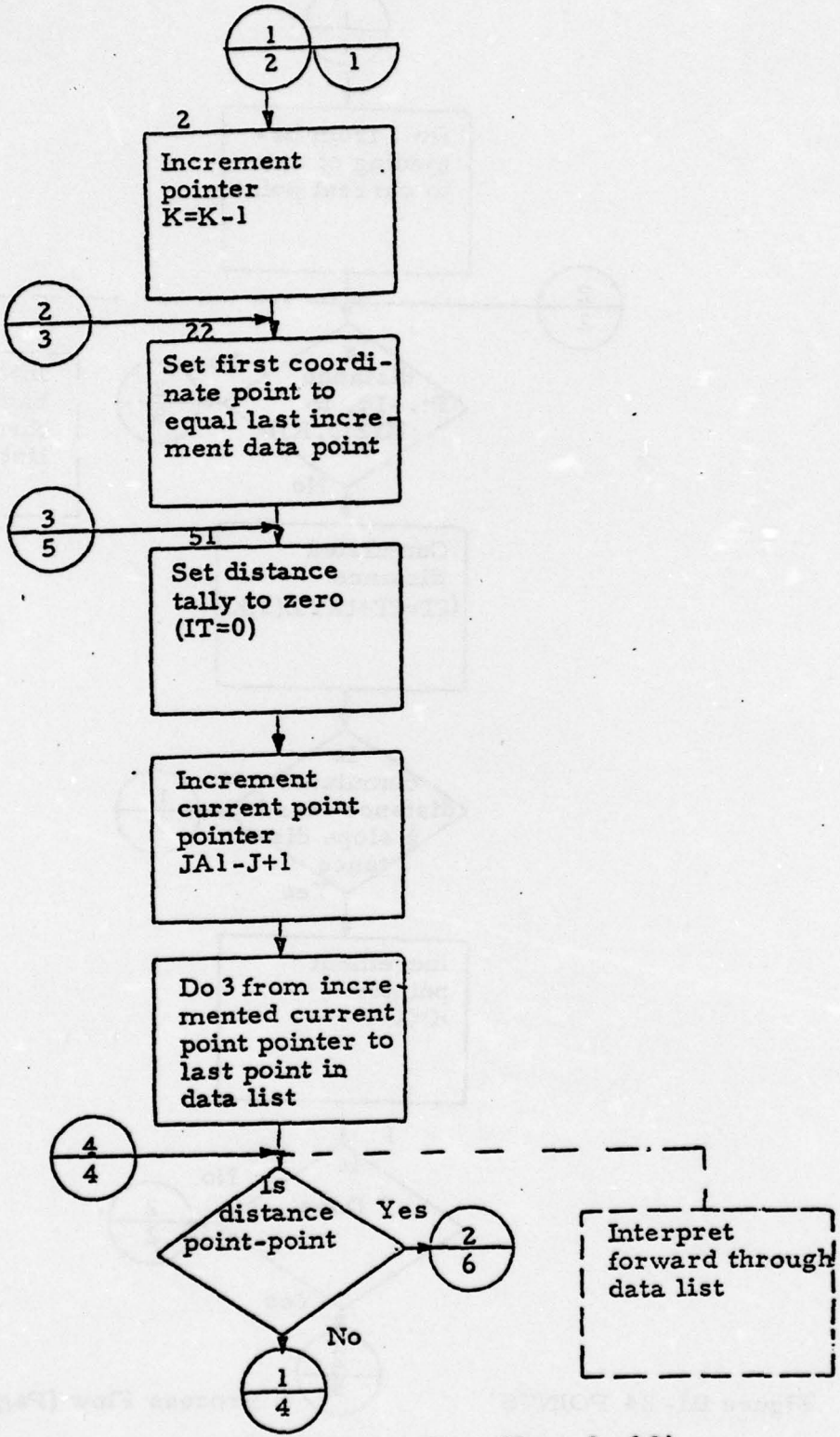



Figure III-24 POINTS Process Flow (Page 3 of 9)

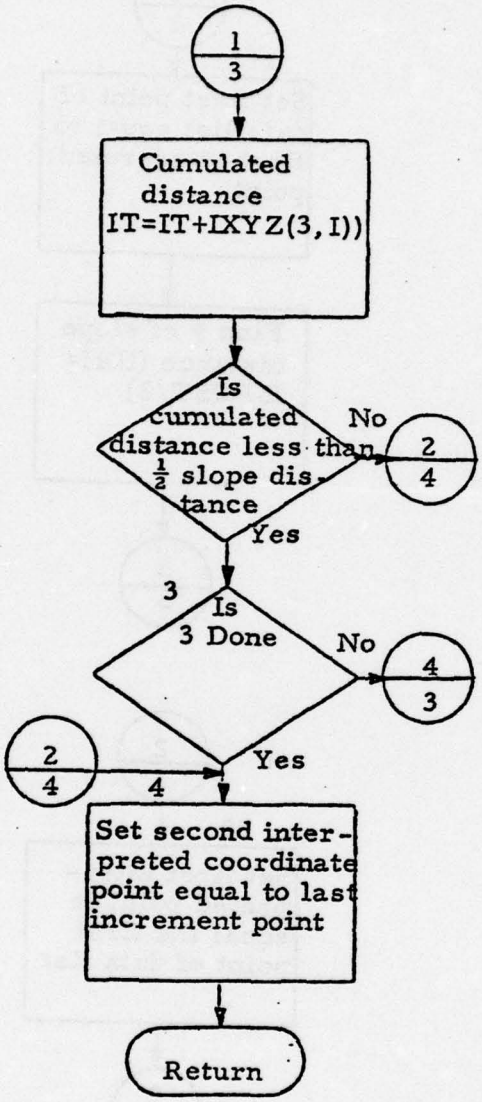



Figure III-24 POINTS Process Flow (Page 4 of 9)

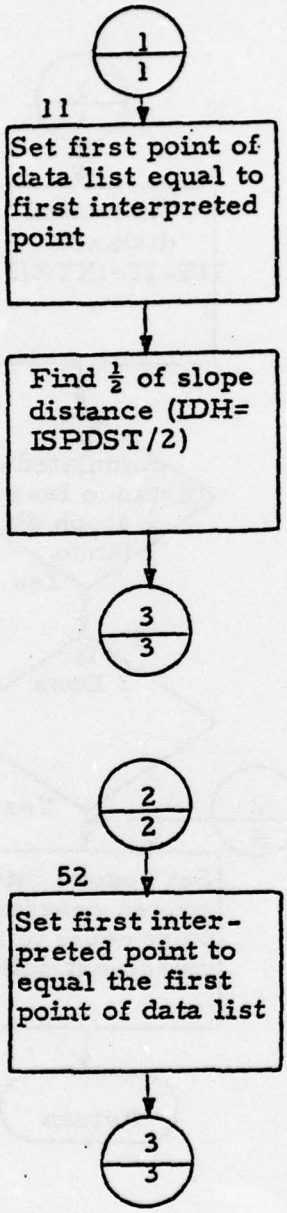



Figure III-24 POINTS Process Flow (Page 5 of 9)

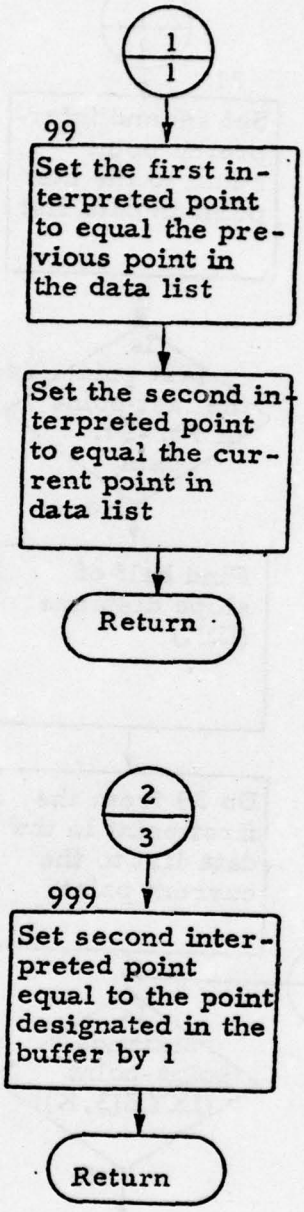



Figure III-24 POINTS Process Flow (Page 6 of 9)

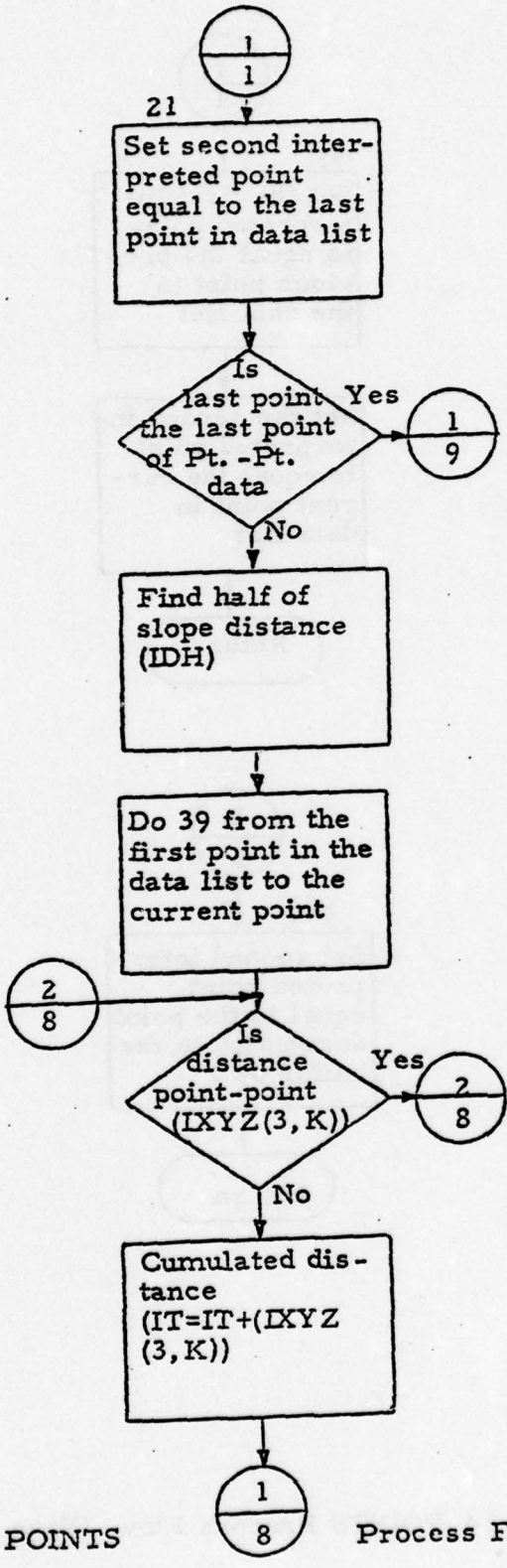



Figure III-24 POINTS Process Flow (Page 7 of 9)

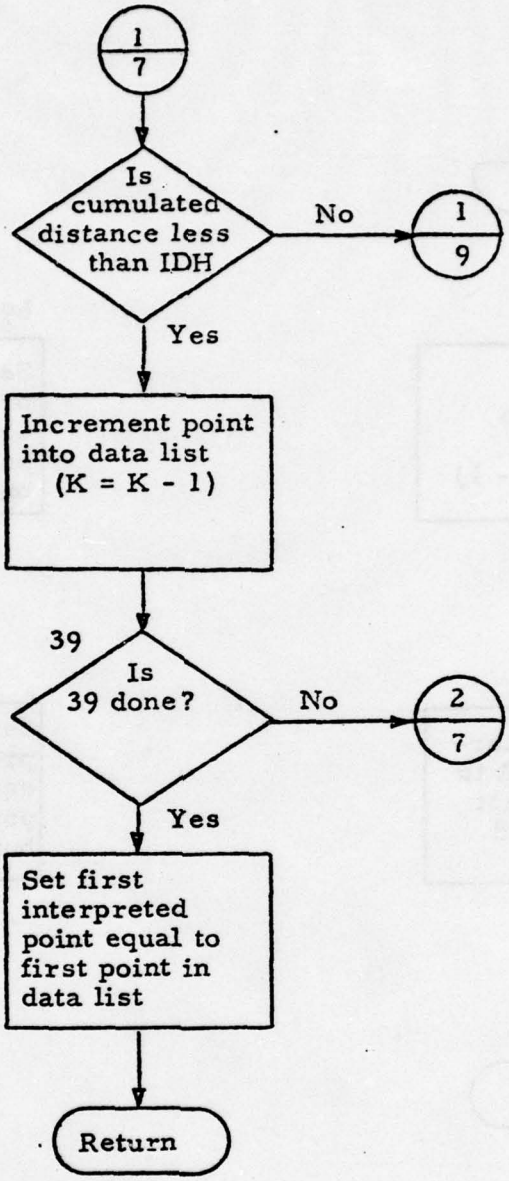



Figure III-24 POINTS Process Flow (Page 8 of 9)

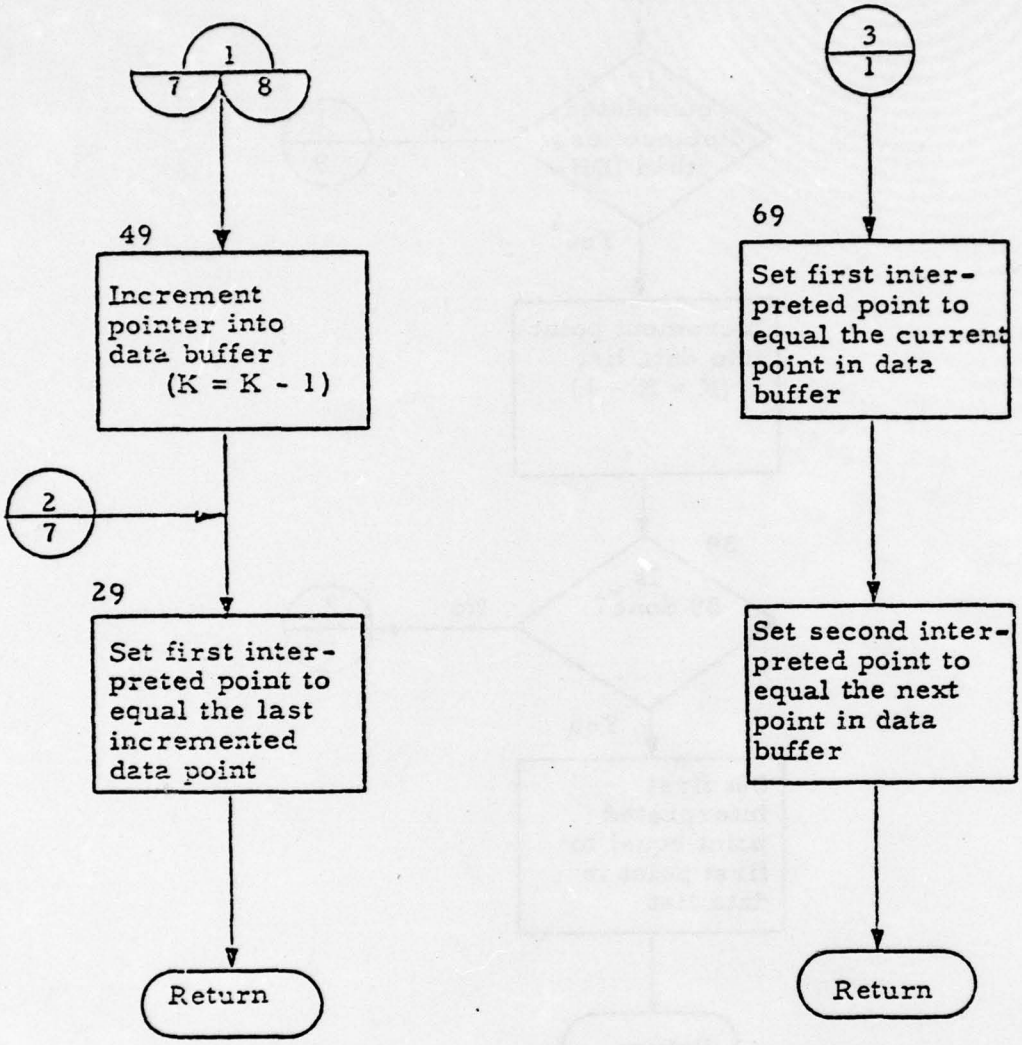



Figure III-24 POINTS Process Flow (Page 9 of 9)

## X. SPACE

### 1. Functional Description

The primary processing task of subroutine SPACE is to generate a space segment along a lineal feature data string.

### 2. Computer Definition

#### a. Core Memory Used

202 octal words.

#### b. Peripheral Equipment

Not applicable

### 3. Program Description

#### a. Calling Routine

SIMBOL

#### b. Subroutine Used

FINDPT

#### c. Input

Input consists of data and flags found in the GLSS common area C2 (feature line center data), C3 (symbol specification directives), and C5 (status indicator flags and pointers).

#### d. Output

Output consists of an update buffer index pointer (mnemonic IPTDX) directed by the current index pointer (ICURDX) found in common C2 (feature line center data).

#### e. Processing Methodology

Processing flow of subroutine SPACE is shown in Figure III-25. On entry, the SPACE continuation variable is interrogated.

If false, the symbol ready for output flag (ISYRDY) and the distance variable (IDIST) is cleared. The symbol size is extracted for its respective location (ISYSZ) and placed in mnemonic ISIZE. The number of points processed is calculated (NPTS) and checked against the total number of points in the buffer specified by ICURDX (current index points). If NPTS is less than the total number of points (NUMPTS (ICURDX) ), the distance (IDIST) is checked against the space size (ISIZE). If IDIST is less than the space size the index pointer for the data containing the input coordinates, (IPTDX (ICURDX) ) is incremented. The distance variable (IDIST) is totaled against the next distance in the IXYZ buffer. Control is then passed to the above mentioned distance and points processed check. If the points processed is greater than or equal to the total number of points specified via ICURDX, the tally run out flag (ITELRN (ICURDX) ) is set. When the feature continuation flag is set or the current index pointer is not equal to one, control is returned to the calling routine SIMBOL; otherwise, the SPACE continuation flag is set to two, symbol piece index pointer is saved (ISAVPX), and control is returned to SIMBOL. Upon re-entry, the symbol piece index is reset with process control being passed to the aforementioned distance and points processed checks. When the distance calculation is greater than or equal to the space size, an intermediate point and distance is generated via subroutine FINDPT and stored in the IXYZ buffer with control being returned to the calling routine SIMBOL.

f. Calling Sequence

Call SPACE

g. Major Algorithms

See subroutine FINDPT for the intermediate point and distance calculation.

4. Program Constants and Variables

| | | |
|---|---|---|
| IPTDX | - | current index pointer specified via ICURDX |
| ICURDX | - | current buffer point |
| ISYRDY | - | symbol ready for output |
| IDIST | - | distance variable |
| ISYSZ | - | symbol piece size directive buffer |
| ISIZE | - | symbol piece size variable |
| IXYZ | - | buffer containing the input and output of the coordinates specified by IPTDX (ICURDX) |

5. Error Conditions

None

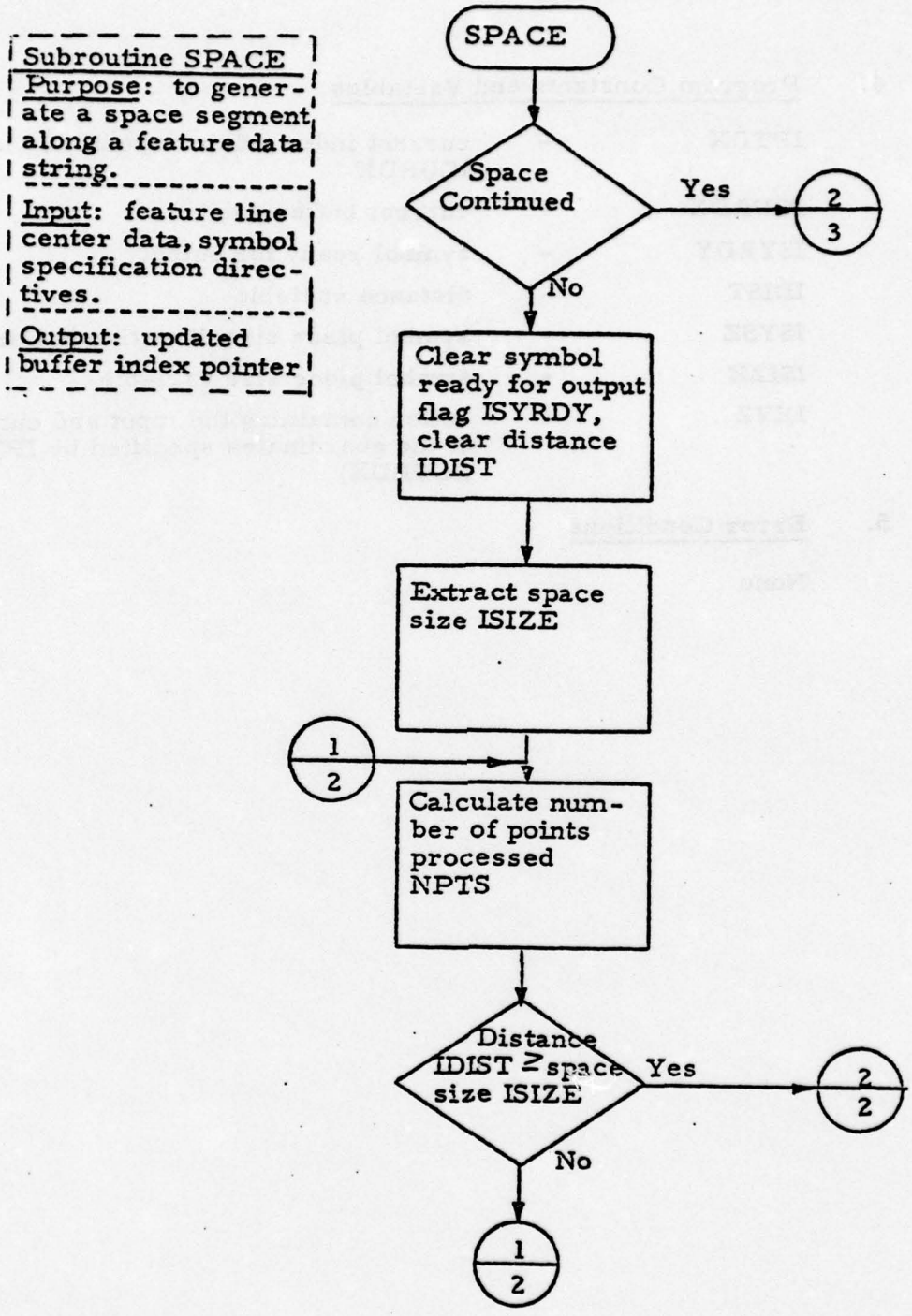



Figure III-25 SPACE Process Flow (Page 1 of 3)

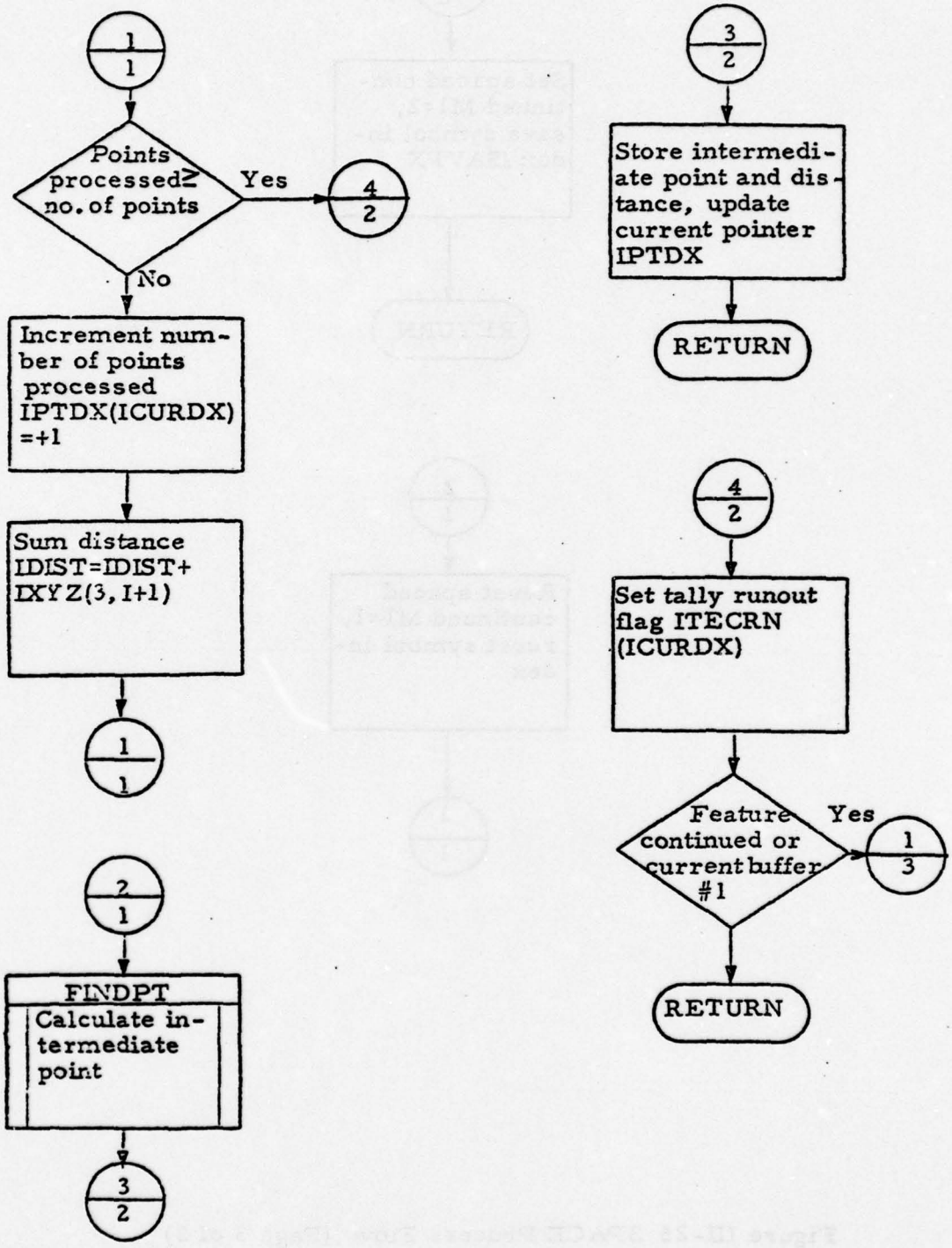



Figure III-25 SPACE Process Flow (Page 2 of 3)

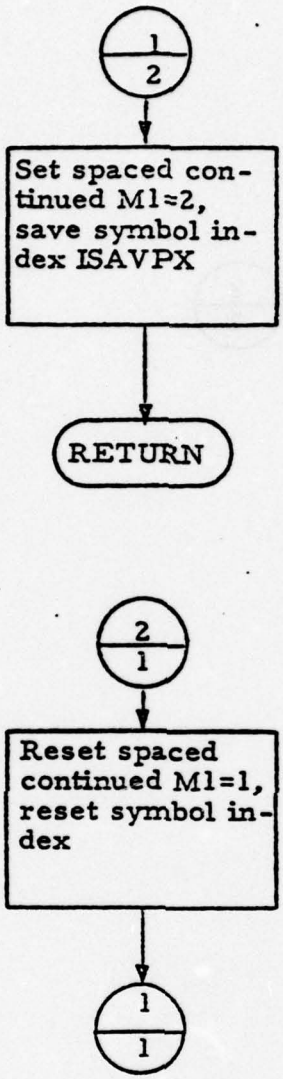



Figure III-25 SPACE Process Flow (Page 3 of 3)

## Y. DOTTER

### 1. Functional Description

The purpose of subroutine DOTTER is to generate a dot symbol at a given coordinate location.

### 2. Computer Definition

#### a. Core Memory Used

73 octal words.

#### b. Peripheral Equipment

Not applicable

### 3. Program Description

#### a. Calling Routine

SIMBOL

#### b. Subroutines Used

None

#### c. Input

The input consists of data found in common area C2 (feature line center data common area) and C5 (status indicators flags and pointers common area).

#### d. Output

The primary output is a coordinate point stored in the output buffer (IXYZ) directed by the current buffer pointer mnemonic ICURDX.

#### e. Processing Methodology

The processing flow of subroutine DOTTER is depicted in Figure III-26. On entry, needed index pointers are initialized, and the

dot counter is incremented (mnemonic NUMDOT). A single coordinate point is extracted from the input buffer, directed via buffer one's index pointer (IPTDX(1) ) and stored into the output buffer, directed by the starting index of buffer sub ICURDX (current index buffer pointer) mnemonic ICORDX (ICURDX). If the number of points processed in buffer one (the first 950 coordinate locations of the buffer IXYZ) is greater than or equal to the number of points in buffer one (NUMPTS(1) ), the tally run flag for buffer one is set (ITELRN(1) ). The symbol ready for output flag is then set with process control being returned to the calling routine SIMBOL.

f. Calling Sequence

Call DOTTER

g. Major Algorithms

None

4. Program Constants and Variables

| | | |
|---|---|---|
| ISYRDY | - | symbol ready for output flag |
| ICURDX | - | current index buffer pointer |
| IPTDX | - | current index pointer of ICURDX |
| ICORDX | - | starting index pointe4s of the five IXYZ buffers |
| NUMDOT | - | number of dots generated |
| NUMPTS | - | number of points in each buffer |
| ITELRN | - | tally run out flags |

5. Error Conditions

None

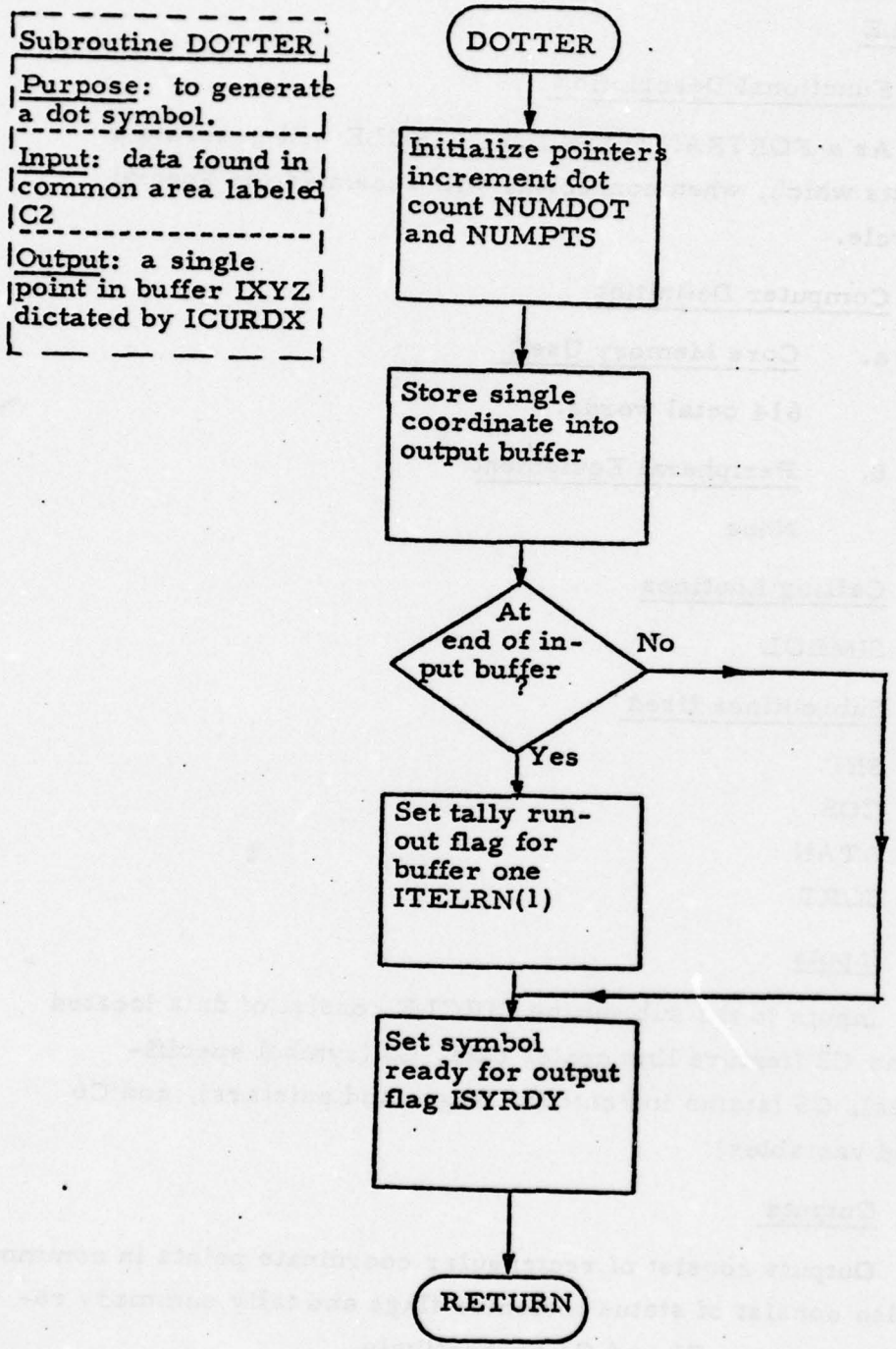



Figure III-26 DOTTER Process Flow (Page 1 of 1)

Z. CIRCLE

1. Functional Description

As a FORTRAN subroutine, CIRCLE will generate a sequence of points which, when connected, will resemble the special point symbol circle.

2. Computer Definition

a. Core Memory Used

614 octal words.

b. Peripheral Equipment

None

3. Calling Routines

SIMBOL

b. Subroutines Used

SIN
COS
ATAN
SQRT

c. Inputs

Inputs to the subroutine CIRCLE consist of data located in common areas C2 (feature line center data, C3 (symbol specification directives), C5 (status indicators, flags, and pointers), and C6 (parameters and variables).

d. Outputs

Outputs consist of rectangular coordinate points in common area C2, and also consist of status indicator flags and tally summary report data in common areas C5 and C7 respectively.

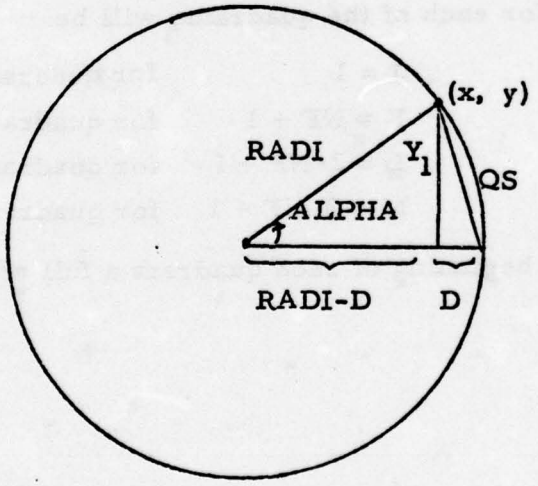
e. Processing Methodology

When called upon, the subroutine CIRCLE will locate the center point in the data list (IXYZ) about which the circle is to be computed. After finding the center point, CIRCLE calculates the incremental angle ALPHA (see Section g: "Major Algorithms"). CIRCLE then finds the rectangular coordinate points for the axis of the circle. From each of the axis points, CIRCLE then increments around, in a counter clockwise direction, until it has completed 90 degrees. At each increment, CIRCLE computes four rectangular coordinate points (see Section g: Major Algorithms) and enters the points in buffer 3 of IXYZ. After completion of the circle and after CIRCLE sets the appropriate flags and indicators to output the coordinate points, control is returned to the calling routine. Refer to Figure III-27 for the process flow diagram of CIRCLE.

f. Calling Sequence

Call CIRCLE

g. Major Algorithms




QS - length of cord
RADI - radius of cord

Using the distance equation and the equation of a circle to find (x, y) with center at (0, 0):

$$1) \quad QS^2 = D^2 + y^2$$

$$2) \quad RADI^2 = (RADI - D)^2 + y^2$$

Subtracting equation 2) from 1) and solving for d:

$$d = QS^2/2R$$

To find y, substitute $QS^2/2R$ for D in equation 1) and solve for y:

$$y = (QS^2 - (QS^2/2R)^{1/2}$$

and solve for x:

$$x = RADI - D$$

To find ALPHA (the incremental angle):

$$ALPHA = ATAN(y/x)$$

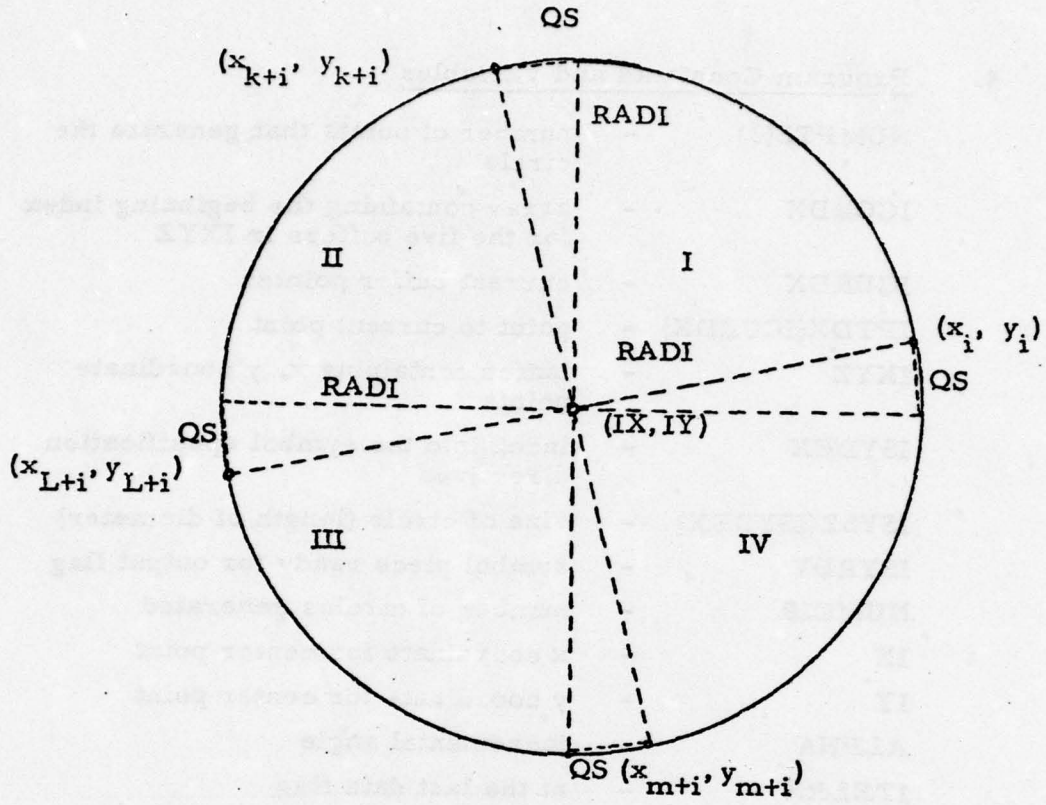
To determine the number of increments (NF) per quadrant:

$$NF = \pi/ALPHA$$

The starting index for each of the quadrants will be:

| | |
|---|---|
| J = 1 | for quadrant 1 |
| K = NF + 1 | for quadrant 2 |
| L = 2·NF + 1 | for quadrant 3 |
| M = 3·NF + 1 | for quadrant 4. |

Increment from the beginning of each quadrant a full $\pi/4$ such that:




i = 1 to NF

IC = RADI * COS (i*ALPHA)

IS = RADI * SIN (i * ALPHA)

$x_i = IX + IC$

$y_i = IY + IS$

$x_{k+i} = IX - IS$

$y_{k+i} = IY + IC$

$x_{L+i} = IX - IC$

$y_{L+i} = IY - IS$

$x_{m+i} = IX + IS$

$y_{m+i} = IY - IC$

4. Program Constants and Variables

| | | |
|---|---|---|
| NUMPTS(3) | - | number of points that generate the circle |
| ICORDX | - | array containing the beginning index for the five buffers in IXYZ |
| ICURDX | - | current buffer pointer |
| IPTDX(ICURDX) | - | point to current point |
| IXYZ | - | buffer containing x, y coordinate points |
| ISYDEX | - | index into the symbol specification directives |
| ISYSZ(ISYDEX) | - | size of circle (length of diameter) |
| ISYRDY | - | symbol piece ready for output flag |
| NUMCIR | - | number of circles generated |
| IX | - | x coordinate for center point |
| IY | - | y coordinate for center point |
| ALPHA | - | incremental angle |
| ITELRN | - | at the last data flag |
| J | - | counter for 1st quadrant of circle |
| K | - | counter for 2nd quadrant of circle |
| L | - | counter for 3rd quadrant of circle |
| M | - | counter for 4th quadrant of circle |
| QS | - | length of cord segment (20.0) |
| IC | - | directional distance of cosine |
| IS | - | directional distance of sin |

5. Error Conditions

None

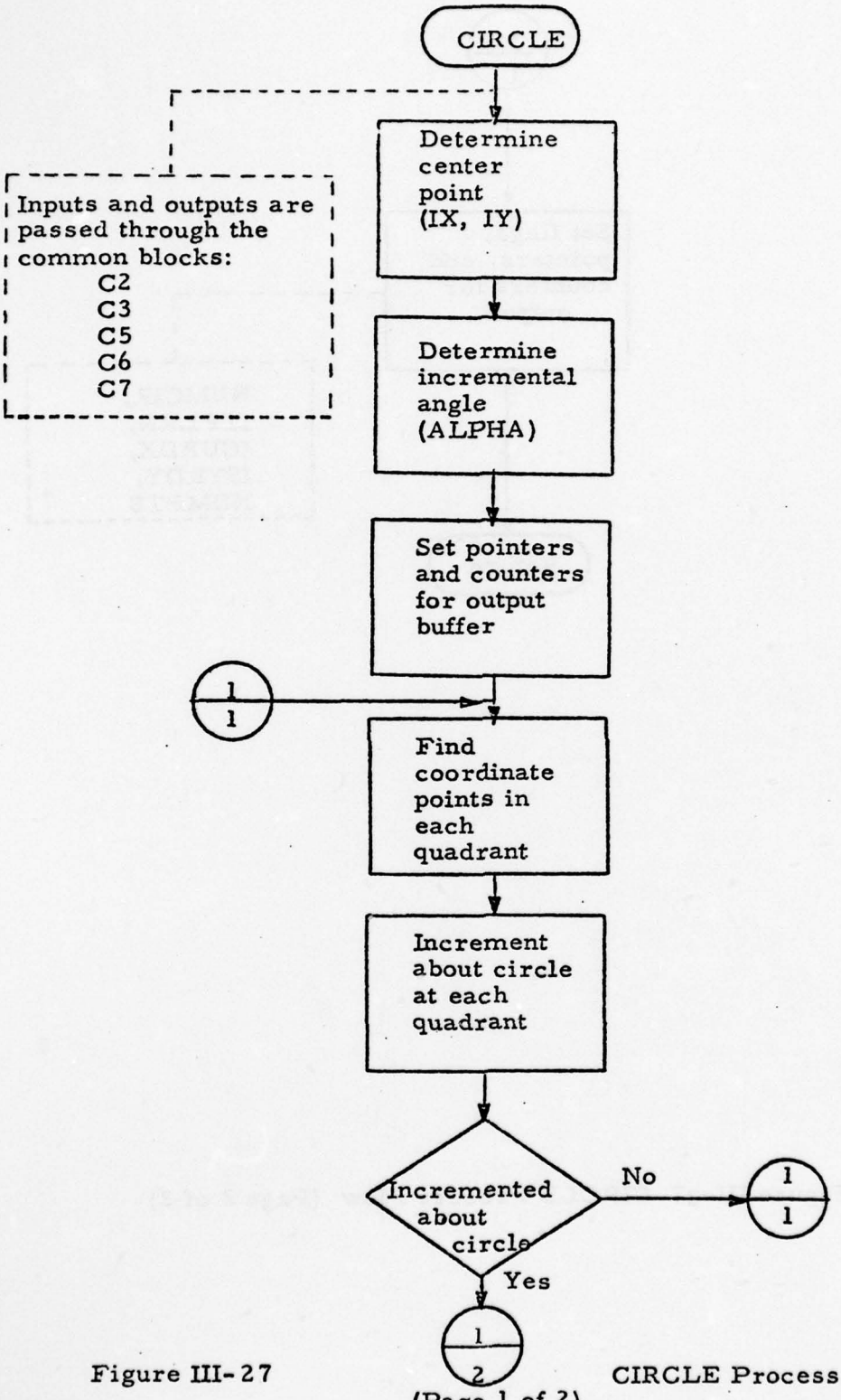



Figure III-27 CIRCLE Process Flow

(Page 1 of 2)

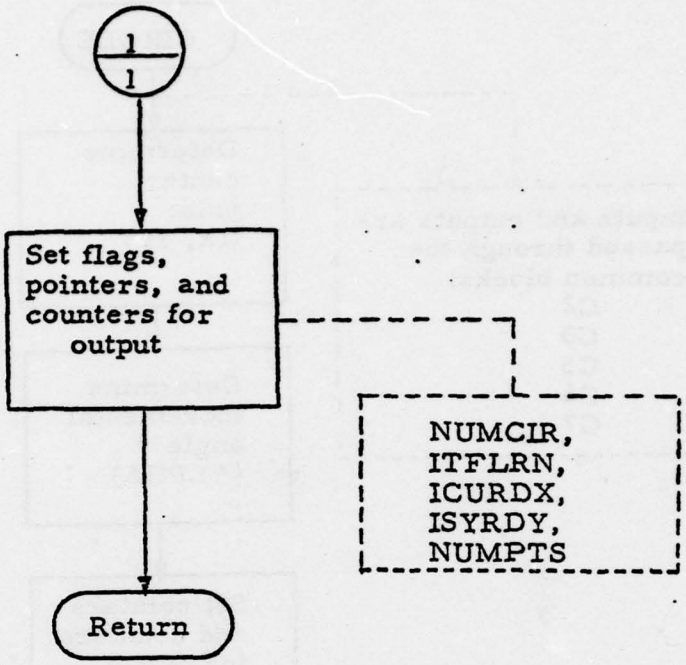



Figure III-27 CIRCLE Process Flow (Page 2 of 2)

## AA. Subroutine TICKER

### 1. Functional Description

Symbolization subroutine TICKER's major processing tasks are to generate feature symbology of full tick marks and half tick marks along a given input feature. A full tick mark is a straight line of short length that intersects a feature line segment at perpendicular angles and is also equal distant on each side of the feature line segment. A half tick mark is only one side of the full tick mark and also intersects the line segment at a perpendicular angle.

### 2. Computer Definition

#### a. Core Memory Used

706 octal words.

#### b. Peripheral Equipment

Not Applicable

### 3. Program Description

#### a. Calling Routine

SIMBOL

#### b. Subroutines Used

POINTS
ABSPNT
SQRT

#### c. Input

Primary input consists of data found in the GLSS common area C2 (feature line center data), C3 (symbol specification directives), C5 (status indicator flags and pointers), and C7 (process tally summary report).

d. Output

Output consists of tick symbol coordinate points stored in the output buffer (mnemonic IXYZ) directed by the current index pointer (mnemonic ICURDX). Other output consists of setting or resetting of various flags and pointers located in common area C5 (status indicator flags and pointers), and C7 (process tally summary report).

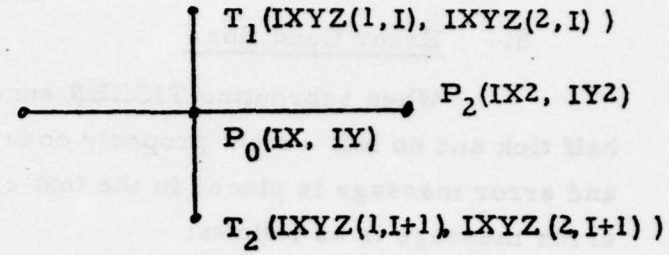
e. Processing Methodology

Processing flow of subroutine TICKER is depicted in Figure III-28. Upon entry, a new feature check is performed, and if found true, the new feature number is saved (menmonic ITPHED). A double tick symbology check is then made with mnemonic IDBLTK being set if double tick symbology is to be performed. The symbol type (ISYTPE) and symbol size (ISIZE) are extracted from their respective symbol specification areas (Common C3). Subroutine POINTS is then called to interpolate over a user input slope distance (mnemonic ISPDST). Subroutine TICKER then calculates an approximate slope distance using x, y coordinate value supplied to it via the above routine. If the double tick symbology flag is set and the symbol index pointer (ISYDEX) is five (second tick), the above slope distance calculation is skipped. If the symbol type is a half tick or alternating half tick, the symbol size is halved. The appropriate tick symbology (tick, half tick, or alternating half tick) x, y coordinates are then calculated with the resultant being stored in the output buffer specified by the current index pointer ICURDX. The tick counter is then incremented by one (menmonic NUMTKS) with the symbol ready for output flag being set (ISYRDY). Process control is returned to the calling subroutine SIMBOL.

f. Calling Sequence

Call TICKER

g. Major Algorithms




| | | |
|---|---|---|
| DIST | = | distance from point $P_1$ to point $P_2$ |
| DIST | = | SQRT (FLOAT ((IX2-IX1)**2 + (IX2-IX1)**2) ) |
| ISIZE2 | = | distance or length of tick (one half of ISIZE for half tick or alternating half tick) |
| SZDDT | = | slope of perpendicular bisect |
| SZDDT | = | FLOAT (ISIZE2) / DIST |
| JX, JY | = | perpendicular bisect translation distances |
| JX | = | (IX1 - IX2) * SZDDT |
| JY | = | (IY2 - IY1) * SZDDT |
| IX, IY | = | point at which tick symbol is to be placed |
| IXYZ (1, I) | = | IX - JY |
| IXYZ (2, I) | = | IY - JX |
| IXYZ (1, I+1) | = | IX + JY |
| IXYZ (2, I+1) | = | IY + JX |

4. Program Constants and Variables

See Common Areas C1 - C7.

5. Error Conditions

When subroutine TICKER encounters feature symbology of half tick and no left - right property code is found (mnemonic LRCODE), and error message is placed in the text error buffer ITXERR. The error message is as follows:

TICKER ERROR-HTICK NO LEFT-RIGHT PROPERTY CODE.

Subroutine TICKER

Purpose: to generate feature symbology of full ticks, half ticks and alternating half ticks.

Input: feature line center data, symbol specification directives, status indicator flags and pointers, and feature descriptor data located in their respective common data buffers.

Output: symbol piece of a full tick, half tick or alternating half tick, store in the proper output buffer via index pointer ICURDX

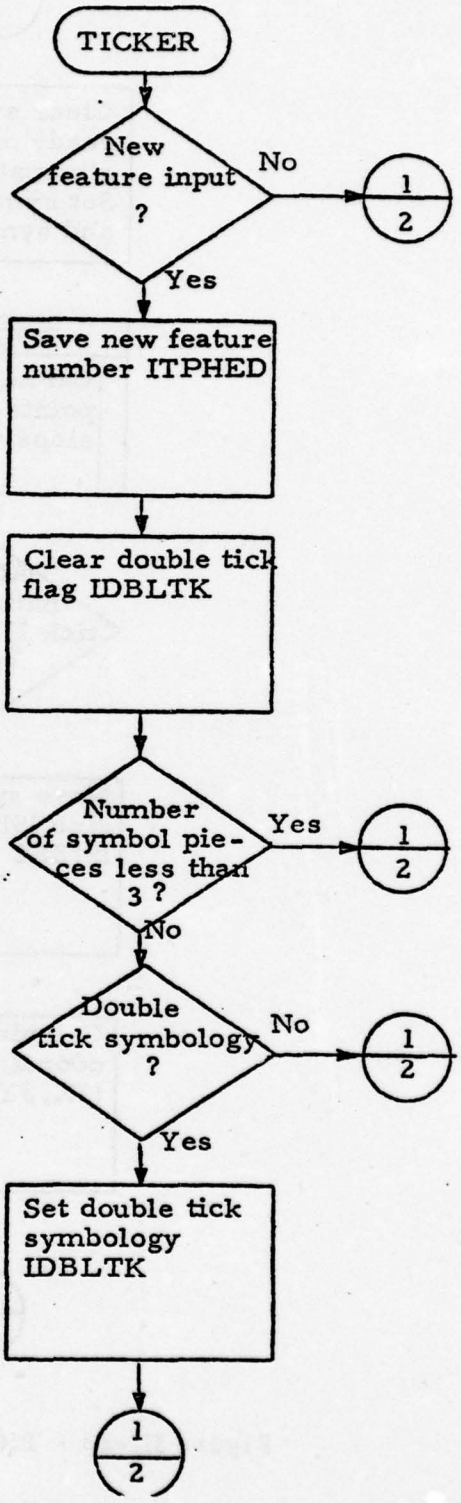



Figure III-28 - TICKER Process Flow (Page 1 of 4)

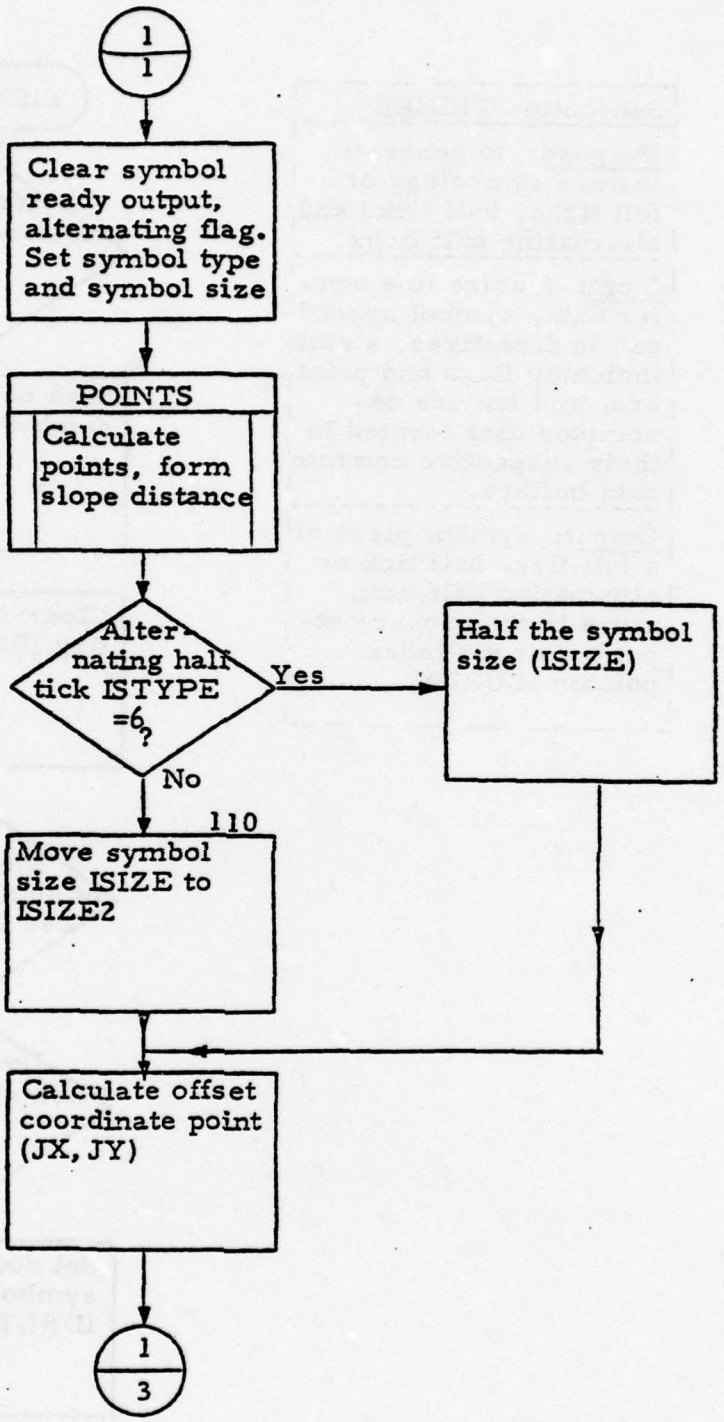



Figure III-28 - TICKER Process Flow
(Page 2 of 4)

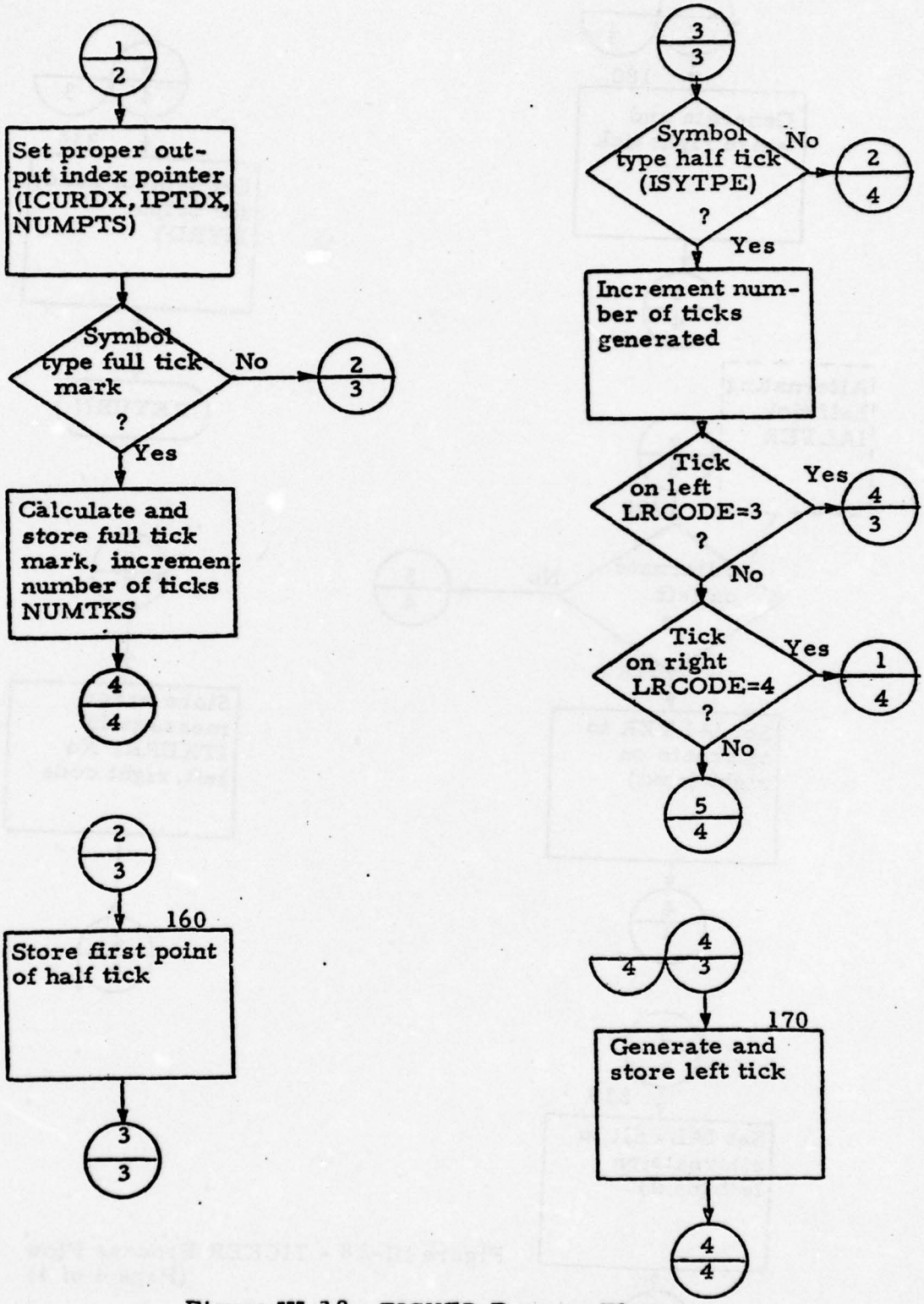



Figure III-28 - TICKER Process Flow
(Page 3 of 4)

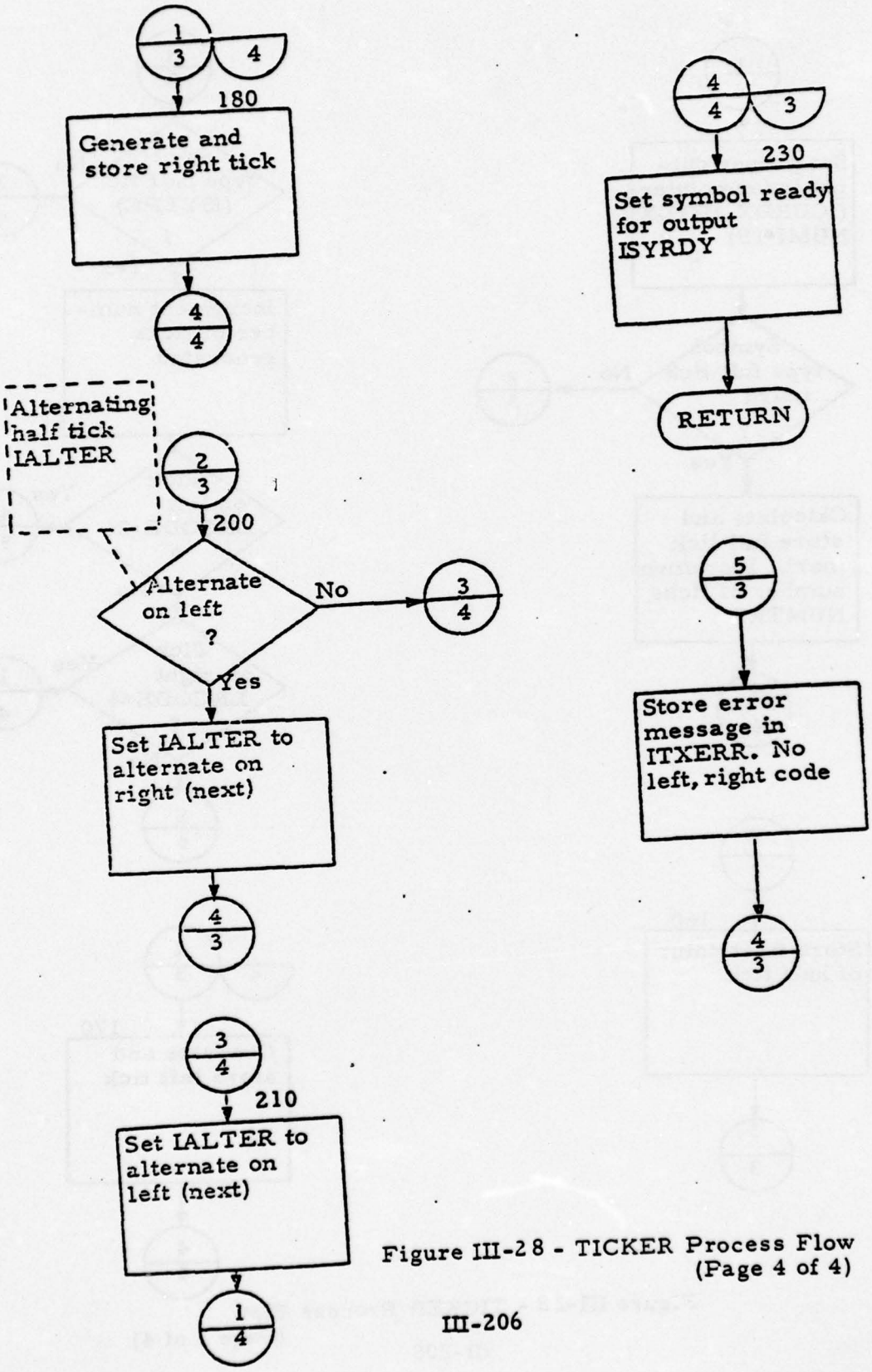



Figure III-28 - TICKER Process Flow
(Page 4 of 4)

BB. CASER

1. Functional Description

As a FORTRAN subroutine, CASER processes a collection of rectangular coordinate points of feature line center data which produces two parallel lines (cased road) about the center line.

2. Computer Definition

a. Core Memory Used

2566 octal words.

b. Peripheral Equipment

None

3. Program Description

a. Calling Routine

SIMBOL

b. Subroutines Used

BAKOVE
CASPOT
BACKUP
CPDIST
CASEIT
CASEST

c. Input

Data as inputs to the subroutine CASER are received through the common areas C2 (feature line center), C3 (symbol specification directives), C5 (status indicators, flags and pointers), and C6 (parameters and variables).

d. Output

Data as output include rectangular data points which are passed in common area C2. Also, outputs include status indicator flags and tally summary report data in common areas C5 and C7, respectively.

e. Processing Methodology

Upon entry to the subroutine CASER, decisions are made to determine the status of the input data, that is, tests are made to see if the internal flags JCONT (casing is to continue after output) and JCALL (CASER was called back to output second side) are set. If either of the two flags (JCONT and JCALL) are set, CASER will proceed accordingly to satisfy the status flag. If the above two flags are not set, a test is made to determine if the total number of data points of the center line feature is greater than one. Any data less than or equal to one presents an error condition which is sent back to the calling routine without any processing on the data. If there is no error condition, processing continues with initializing the data pointers and status flags (IBELL, ICASE, NCL, NCR, NCL1, NCR1, NPTS, NF, NF1, KL, KR). The size of half case length is determined by interrogating the common area C3 for the case size (ISYSZ (ISYDEX)). A test is then made to determine if only two points make up the center line feature, thus requiring casing only at the two end points. A test is then made to determine if the first two points are point-point. Thus, CASER will call CASPOT to case the first point of the feature.

Basically, CASER continues casing, using the following iteration. CASER will increment (NF) through the data testing each distance to determine data which is defined to be point-point. For each increment in the data (trace) which is not point-point, CASER calls

CASEIT followed by BACKUP and CPDIST, or if the flag ICASE is set, BAKOVE is called. For point-point data, CASER calls CASPOT followed by calls to BAKOVE, then sets the flag ICASE. With each call to CASEIT and CASPOT, the left code of the case is stored in buffer three, while the right hand side of the case is stored in buffer four. CASER will perform the above iteration until either the input feature is cased or until buffers three or four are filled.

On the completion of casing the feature, CASER will set the appropriate status flags and update the summary tallies. CASER will return control to the calling routine to output the left side, but CASER will request that control be returned so that the right side will be output.

If buffer three or buffer four have been filled, CASER will output the third buffer then the fourth, then CASER will request control to be returned to it. When control is returned after outputting the fourth buffer, CASER will continue to case as described above. Refer to Figure III-29 for the process flow diagram.

f. Calling Sequence

CALL CASER

g. Major Algorithms

Refer to the algorithms described in:

CASEST
CASEIT
CASPOT
BACKUP
BAKOVE
CPDIST

4. Program Constants and Variables

| | | |
|---|---|---|
| IBELL | - | variable containing current input buffer pointer |
| ICASE | - | flag giving status of point-point casing<br>0 - previous was true data<br>1 - previous was point-point data<br>2 - previous was point-point data followed by trace data |
| ICASEPT | - | flag giving status of first four casing points<br>0 - no point-point data<br>1 - contained point-point data |
| ICORDX | - | array giving the beginning index for the five buffers in IXYZ |
| ICURDX | - | pointer of the current buffer of IXYZ |
| ID | - | calculated distance between two case points |
| IMAXDT | - | maximum distance to be considered trace data |
| IMINDT | - | minimum distance between two points which is accepted |
| IPECLK | - | symbol piece call back flag |
| ISYDEX | - | current symbol piece pointer |
| ISYRDY | - | symbol ready for output flag |
| ISYTP(ISYDEX) | - | size of case |
| ITELRN | - | tally run out flag, i.e., reach the last point at the end of current buffer |
| ITXERR | - | array containing error text messages |
| IXYZ | - | two dimension array containing x, y coordinates of both input and output data |
| JCALL | - | internal flag to CASER to output second side |
| JCONT | - | internal flag to CASER telling CASER to continue casing after output upon reentry. |

| | | |
|---|---|---|
| NCL | - | current pointer into IXYZ for left side of case |
| NCL1 | - | NCL + 1 |
| NCL2 | - | NCL + 2 |
| NCLM1 | - | NCL - 1 |
| NCLM2 | - | NCL - 2 |
| NCR | - | current pointer into IXYZ for right side of case |
| NCR1 | - | NCR + 1 |
| NCR2 | - | NCR + 2 |
| NCRM1 | - | NCR - 1 |
| NCRM2 | - | NCR - 2 |
| NF | - | current pointer at the center line data in IXYZ buffer to be cased |
| NF1 | - | NF + 1 |

5. Error Conditions

CASER will set the error flag and enter the following message into the error text location (ITXERR) if only one point has been input to be cased.

NO DATA OR ONLY ONE DATA POINT TO BE CASE.

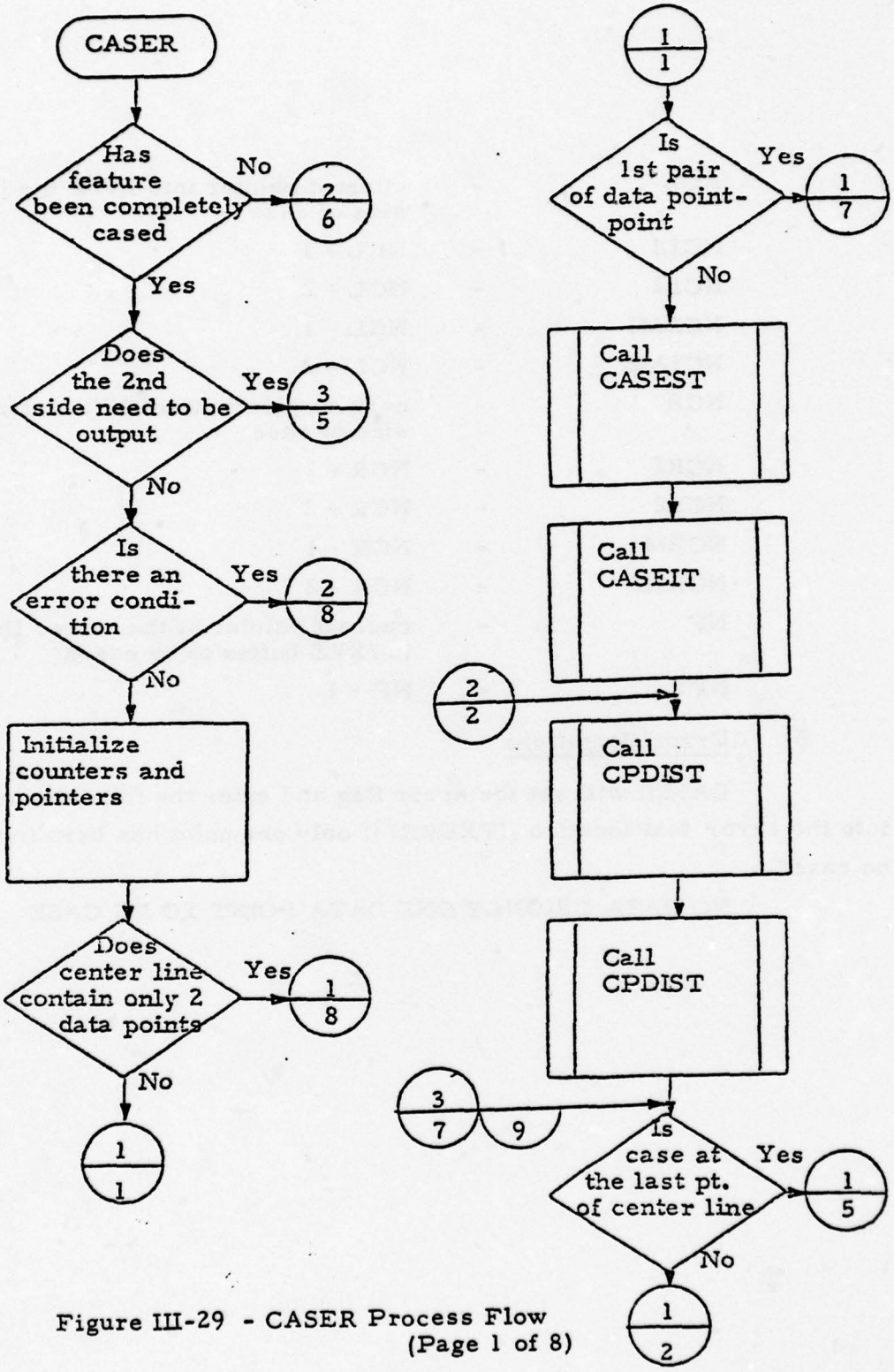



Figure III-29 - CASER Process Flow
(Page 1 of 8)

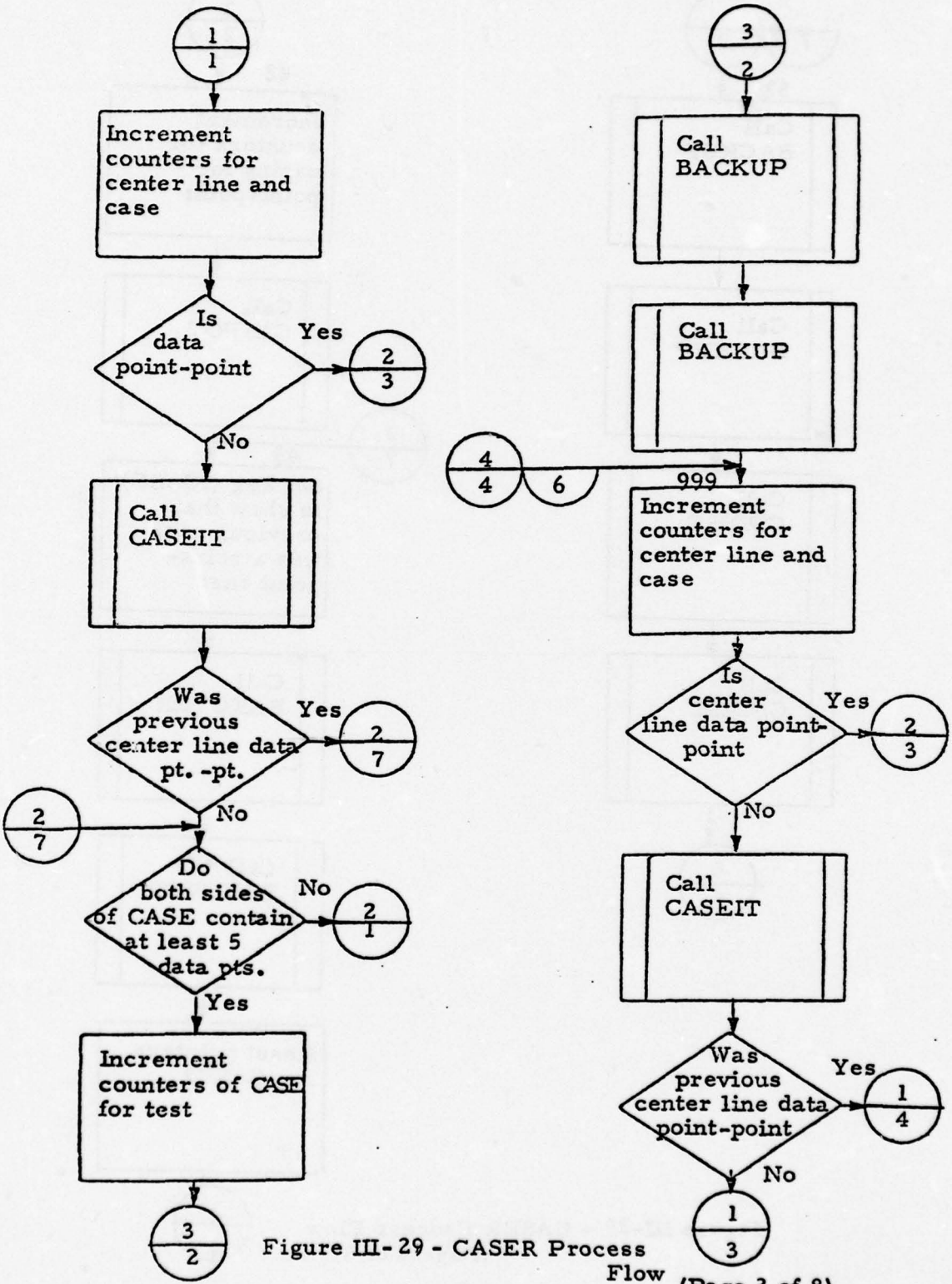



Figure III-29 - CASER Process Flow (Page 2 of 8)

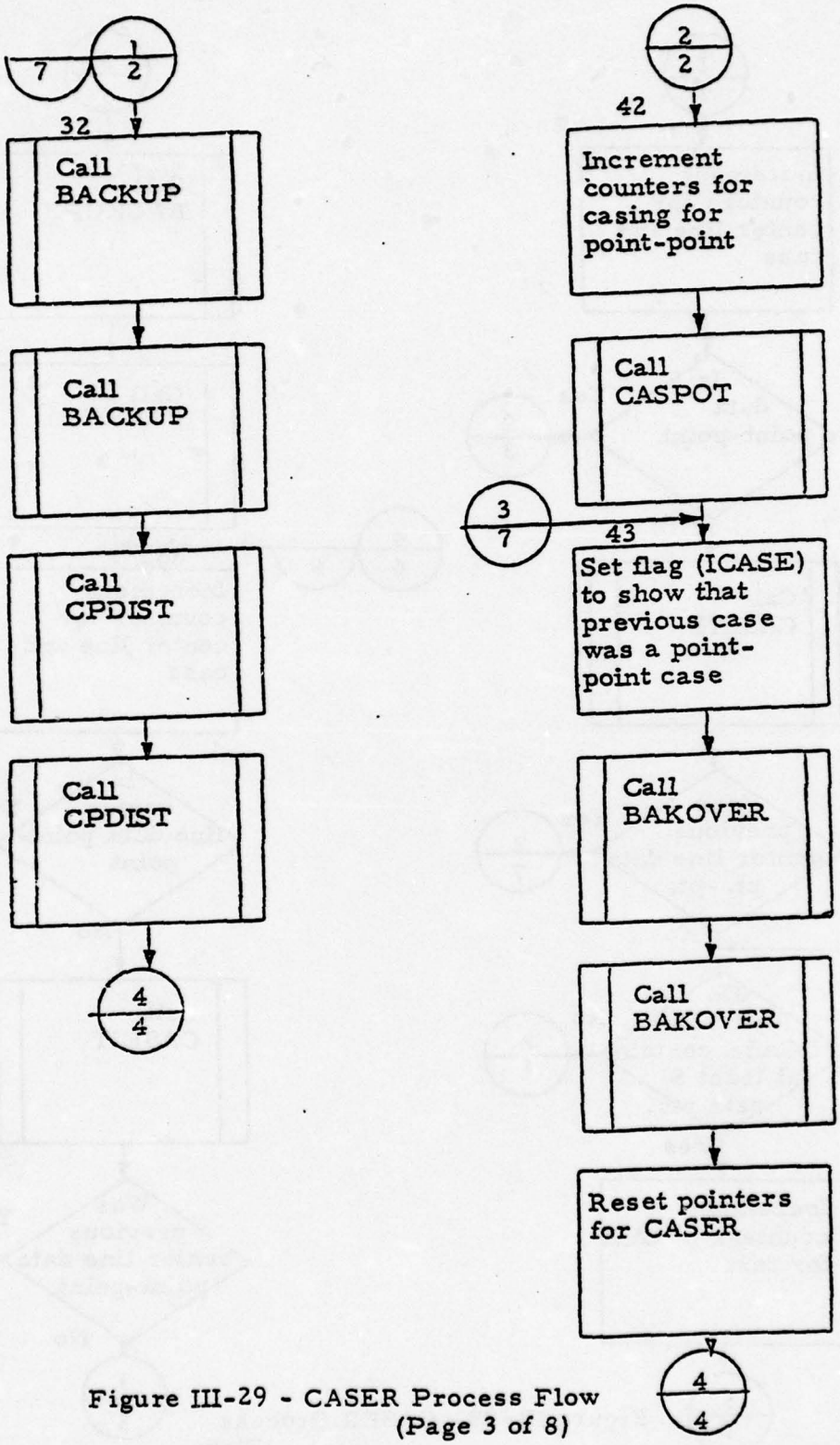



Figure III-29 - CASER Process Flow
(Page 3 of 8)

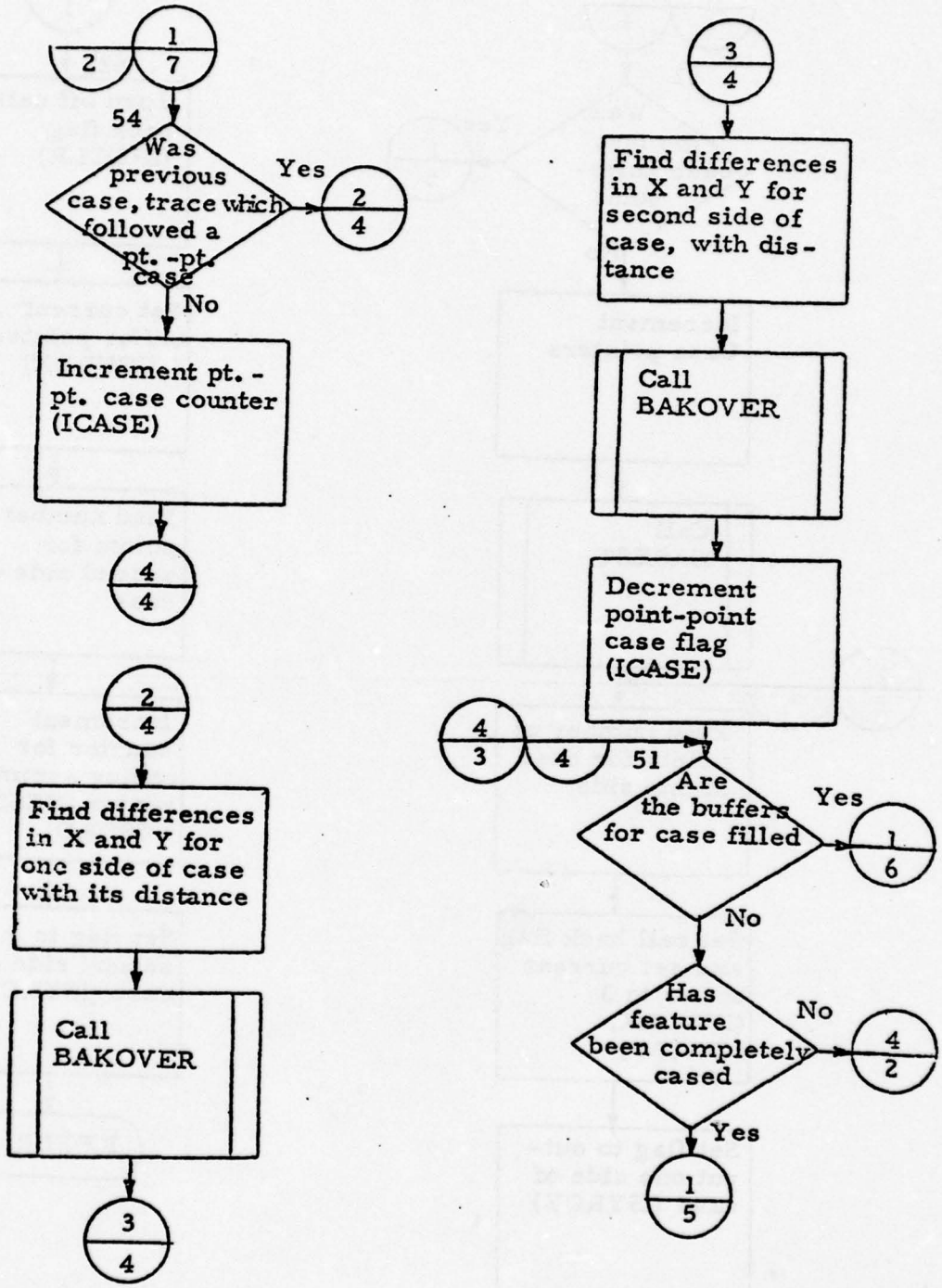



Figure III-29 - CASER Process Flow
(Page 4 of 8)

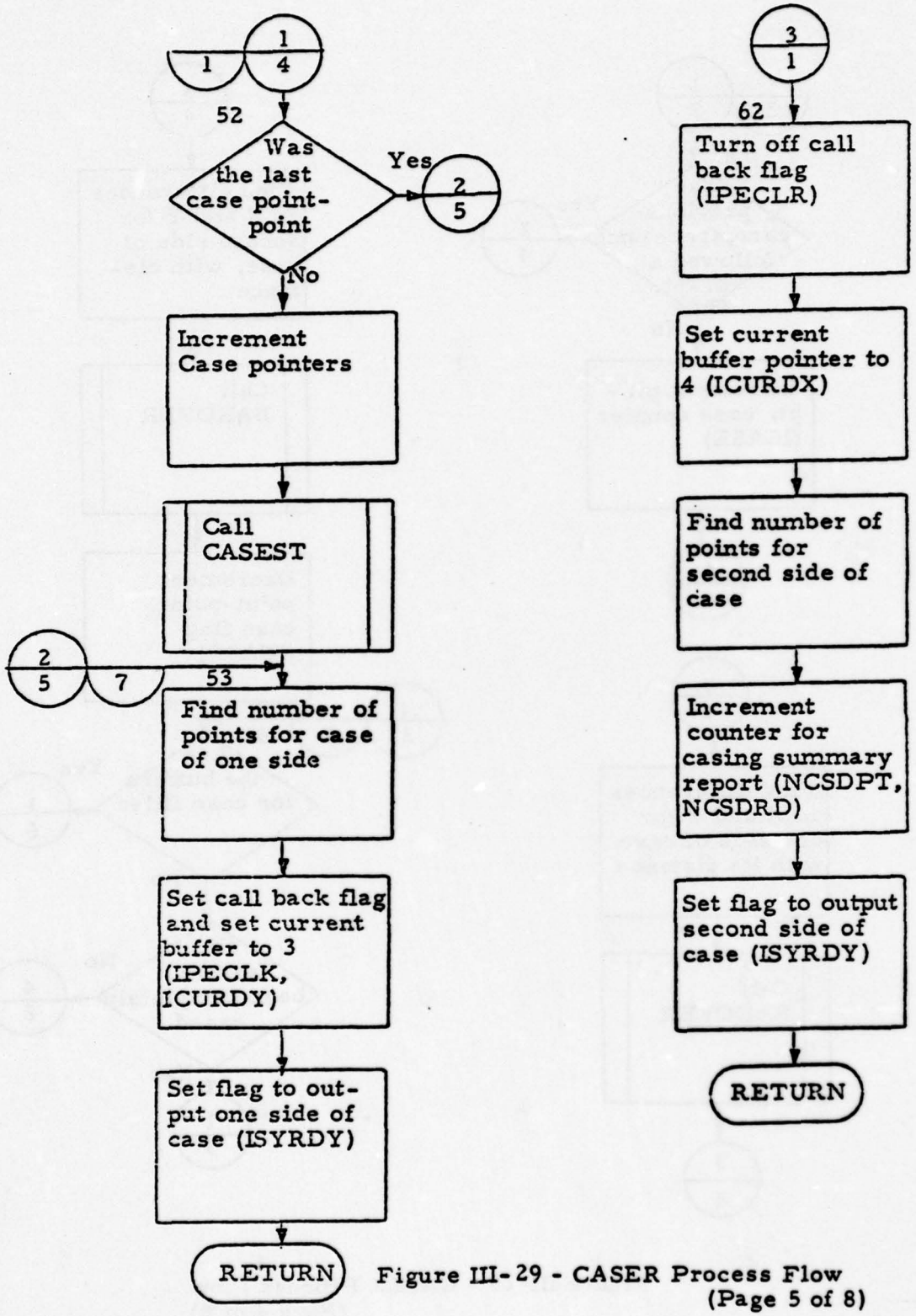



Figure III-29 - CASER Process Flow
(Page 5 of 8)

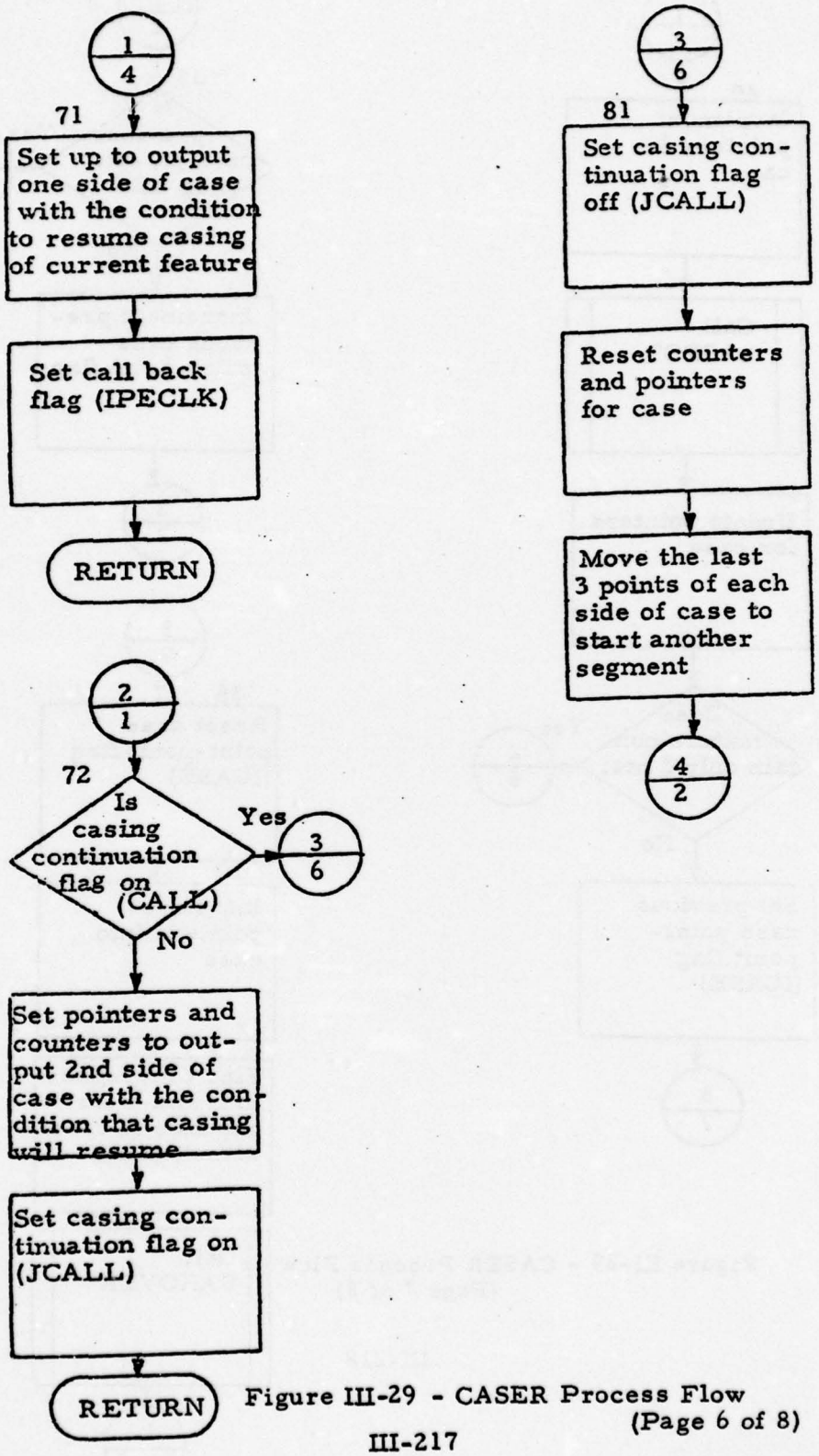



Figure III-29 - CASER Process Flow
(Page 6 of 8)

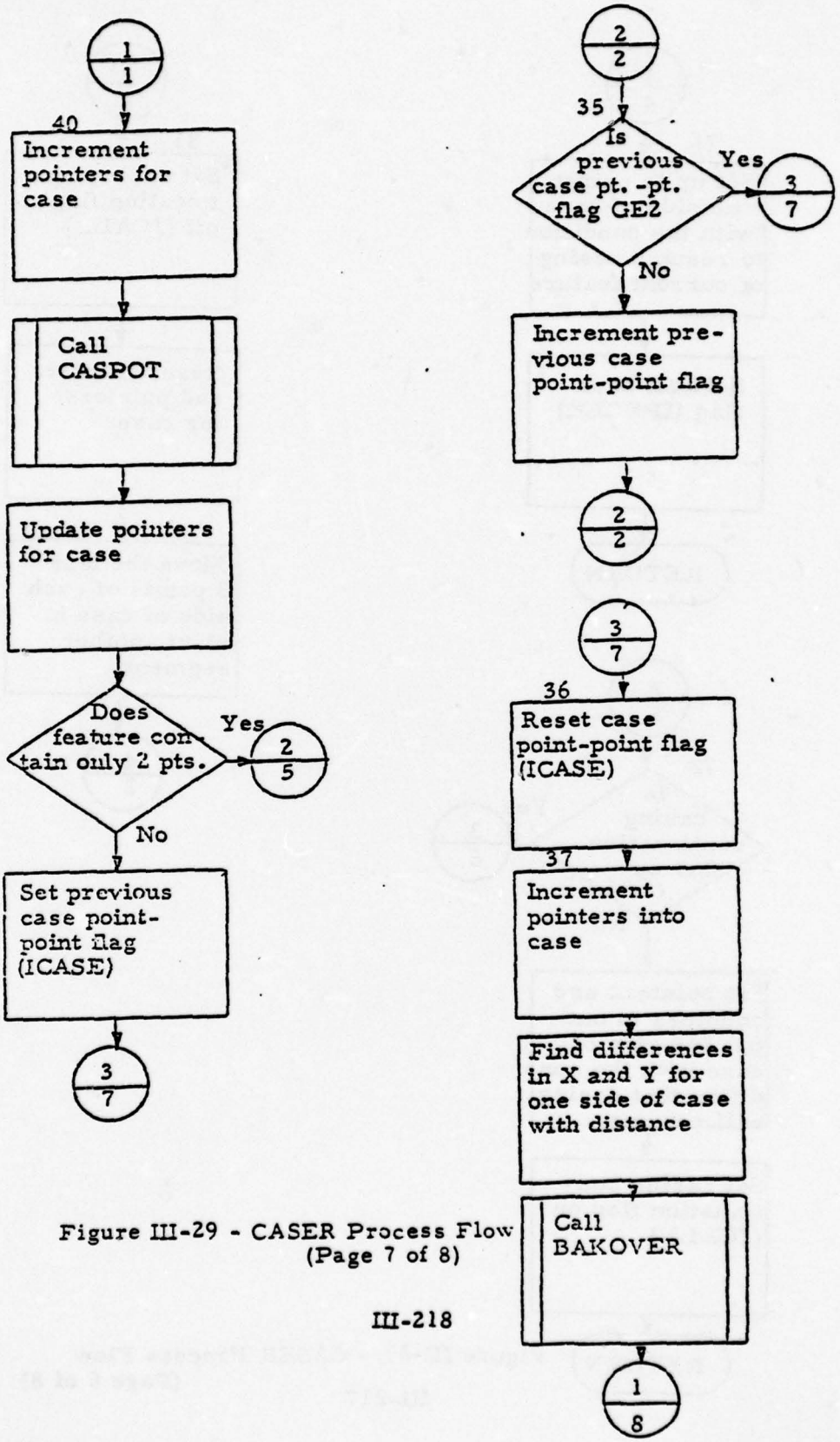



Figure III-29 - CASER Process Flow
(Page 7 of 8)

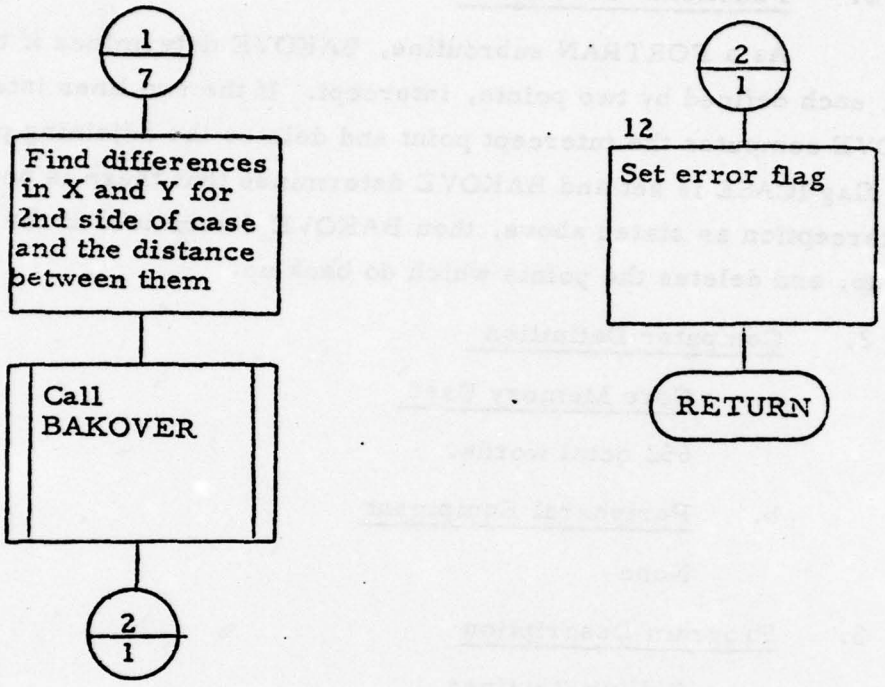



Figure III-29 - CASER Process Flow

(Page 8 of 8)

## CC. BAKOVE

### 1. Functional Description

As a FORTRAN subroutine, BAKOVE determines if two lines, each defined by two points, intercept. If the two lines intercept, BAKOVE computes the intercept point and deletes the adjoining points. If the flag ICASE is set and BAKOVE determines that there is not an interception as stated above, then BAKOVE determines if the points back up, and deletes the points which do back up.

### 2. Computer Definition

#### a. Core Memory Used

652 octal words.

#### b. Peripheral Equipment

None

### 3. Program Description

#### a. Calling Routines

CASER

#### b. Subroutines Used

None

#### c. Input

Inputs are received through the argument list.

| | |
|---|---|
| IX1, IY1 | - x, y coordinates for the first point |
| IX2, IY2 | - x, y coordinates for the second point |
| IX3, IY3 | - x, y coordinates for the third point |
| IX4, IY4 | - x, y coordinates for the fourth point |
| N | - index of the third point |
| N1 | - index of the fourth point |

ICASE - flag indicating if BAKOVE is to test for points that back up, if no interception;
if ICASE = 1, test
if ICASE ≠ 1, no test is to be done

d. Output

Outputs are returned through the argument list. If BAKOVE finds an intercept point, the coordinates of that point are entered into the coordinates of the second point (IX2, IY2), and the coordinates of the fourth point (IX4, IY4) are moved into the coordinates of the third point (IX3, IY3), which in effect deletes the third point. To tell the calling routine that an intercept point was computed, the index of the fourth point (N1) is set equal to the index of the third point (N).

e. Processing Methodology

As a subroutine, BAKOVE, when called upon, will first determine whether the flag ICASE is on or off (1 or 0 respectively). If ICASE is on, BAKOVE proceeds with finding the intercept point, as described below. If ICASE is off, BAKOVE will determine the possibility of an interception by looking at the four points which are received through the argument list as three separate lines. A comparison of distances is made to determine if there is a shorter route (distance) from the first point to the last point, rather than following the four points in order (see the following algorithm). If a shorter route is not determined, BAKOVE returns control to the calling routine. If a shorter distance was found or ICASE is off, BAKOVE continues with finding the intercept point (IX, IY) (refer to the following algorithm). After finding the intercept point, BAKOVE moves the coordinates of the intercept point and the coordinates of the fourth point into the location of the coordinates of the second and third point respectively. The index of the fourth point (N1) is reduced to the index of the third point(N). The above, in fact, removes the adjoining points and inserts the intercept point. Control is then returned to the calling routine. Refer to Figure III-30 for the process flow diagram.

f. Calling Sequence

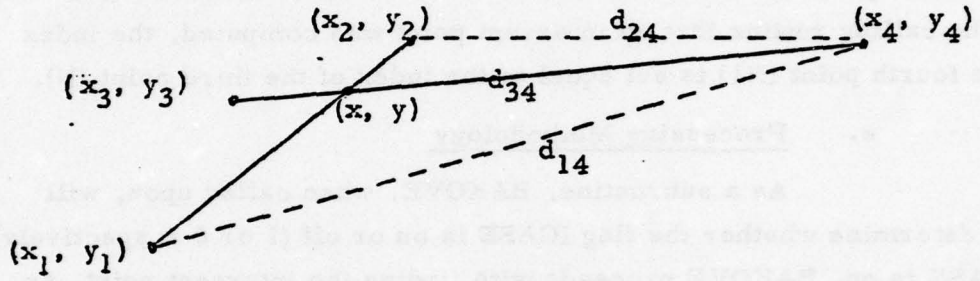
Call BAKOVE (IX1, IY1, IX2, IY2, IX3, IY3, IX4, IY4, IDVAL, N, N1, ICASE).

g. Major Algorithms

The following tests determine the existence of a relative intercept point.

Let $d_{ij}$ = distance from point i to point j.

For ICASE = 0,



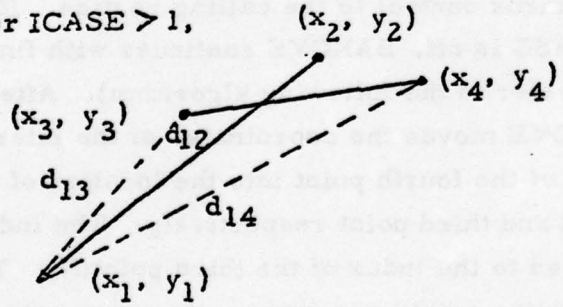

the following must be satisfied,

$$d_{14} < d_{34}$$

or

$$d_{24} < d_{34}.$$

For ICASE > 1,



the following must be satisfied,

$$d_{13} < d_{12}$$

or

$$d_{14} < d_{12} .$$

If either of the two cases above are satisfied, then the following condition must also be satisfied.

Let IS1 be the sign of the slope from the first point $(x_1, y_1)$ to the second point $(x_2, y_2)$,

and

let IS2 be the sign of the slope from the third point $(x_3, y_3)$ to the fourth point $(x_4, y_4)$.

If IS1 $\neq$ IS2, a relative intercept point exists.

Now the intercept can be found by solving the simultaneous equations for x and y.

$$y = m_1 x + b_1$$
$$y = m_2 x + b_2$$

where

$$m_1 = (y_2 - y_1) / (x_2 - x_1)$$
$$m_2 = (y_4 - y_3) / (x_4 - x_3)$$

and

$$b_1 = y_1 - m_1 \cdot x_1$$
$$b_2 = y_3 - m_2 \cdot x_3$$

4. Program Constants and Variables

| | | |
|---|---|---|
| B | - | Y intercept of the first line |
| B1 | - | Y intercept of the second line |
| ICASE | - | flag defining location of intercept point |

| | | |
|---|---|---|
| IDAR | - | distance for testing |
| IDAR1 | - | second distance for testing |
| IDVAL | - | distance received from calling routine |
| IS1 | - | the sign of the slope for the first line |
| IS2 | - | the sign of the slope for the second line |
| IX, IY | - | coordinates for the intercept point |
| IX1, IY1 | - | coordinates of the first point received from the calling routine |
| IX2, IY2 | - | coordinates of the second point received from the calling routine |
| IX3, IY3 | - | coordinates of the third point received from the calling routine |
| IX4, IY4 | - | coordinates of the fourth point received from the calling routine |
| N | - | index for the third point |
| N1 | - | index for the fourth point |
| SLOPE | - | slope of the first line |
| SLOPE1 | - | slope of the second line |

5. Error Conditions

None

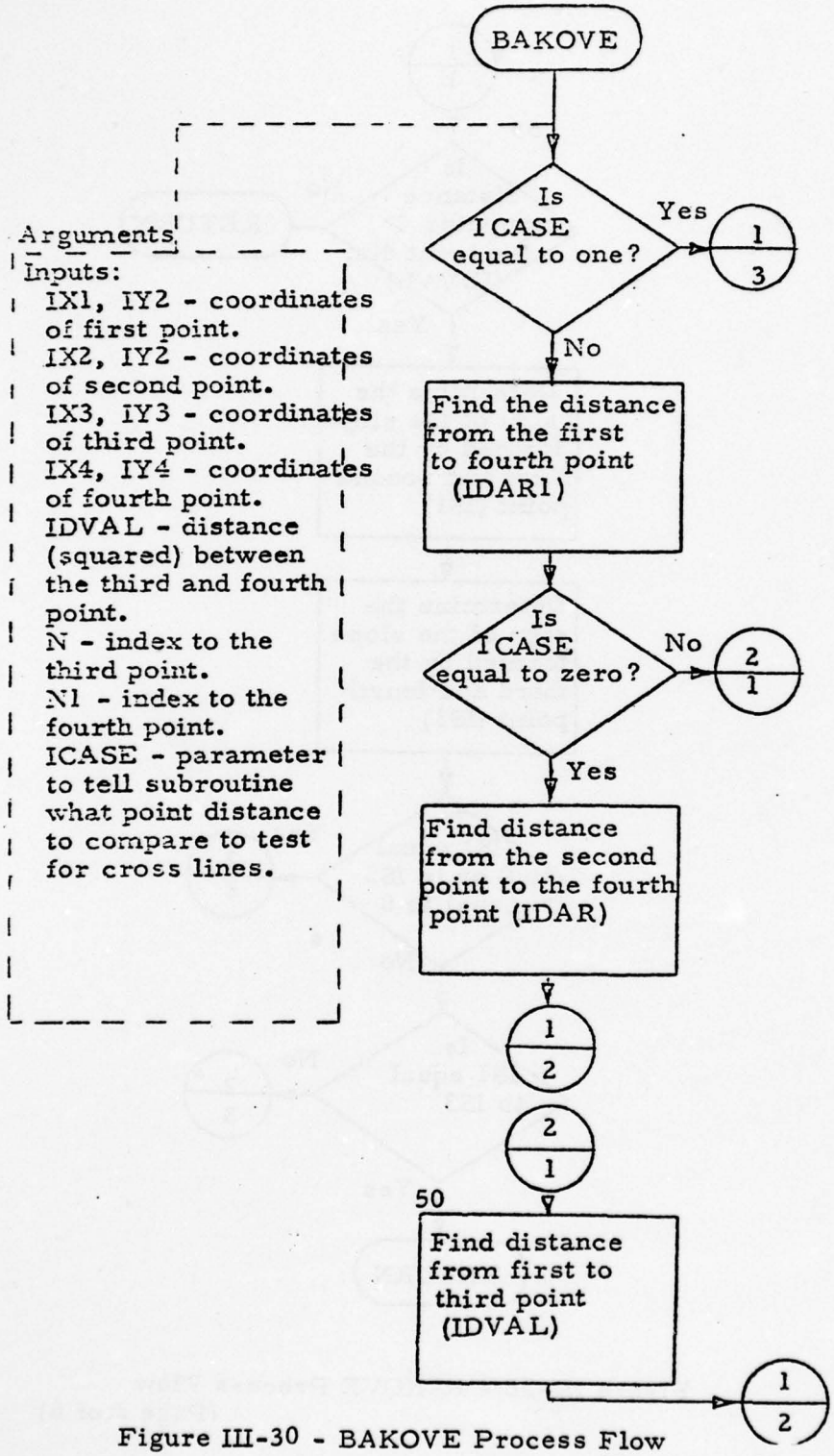



Figure III-30 - BAKOVE Process Flow (Page 1 of 6)

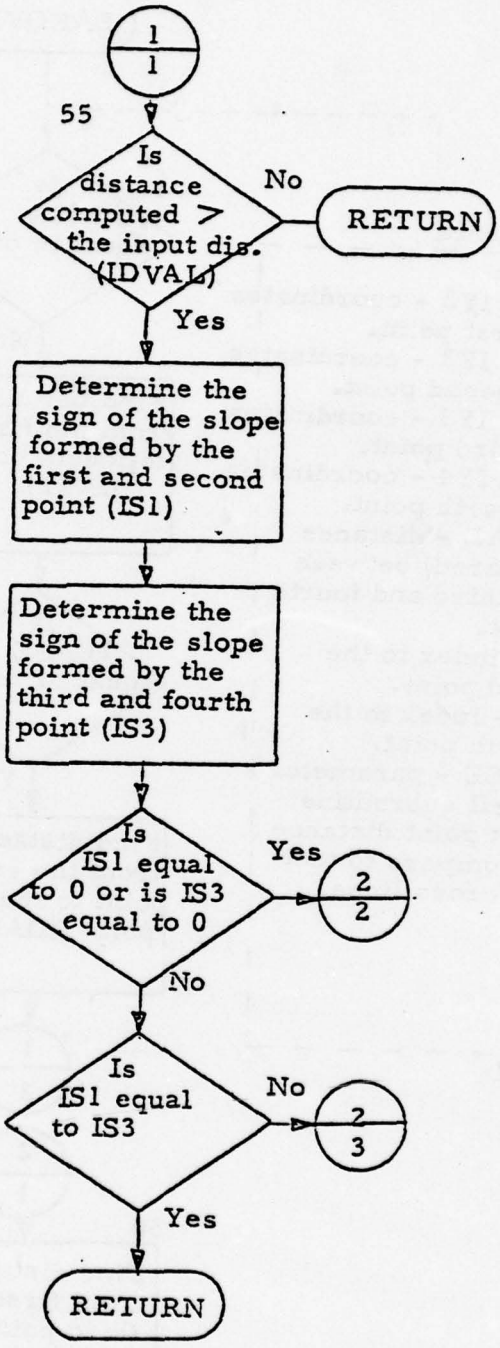



Figure III-30 - BAKOVE Process Flow
(Page 2 of 6)

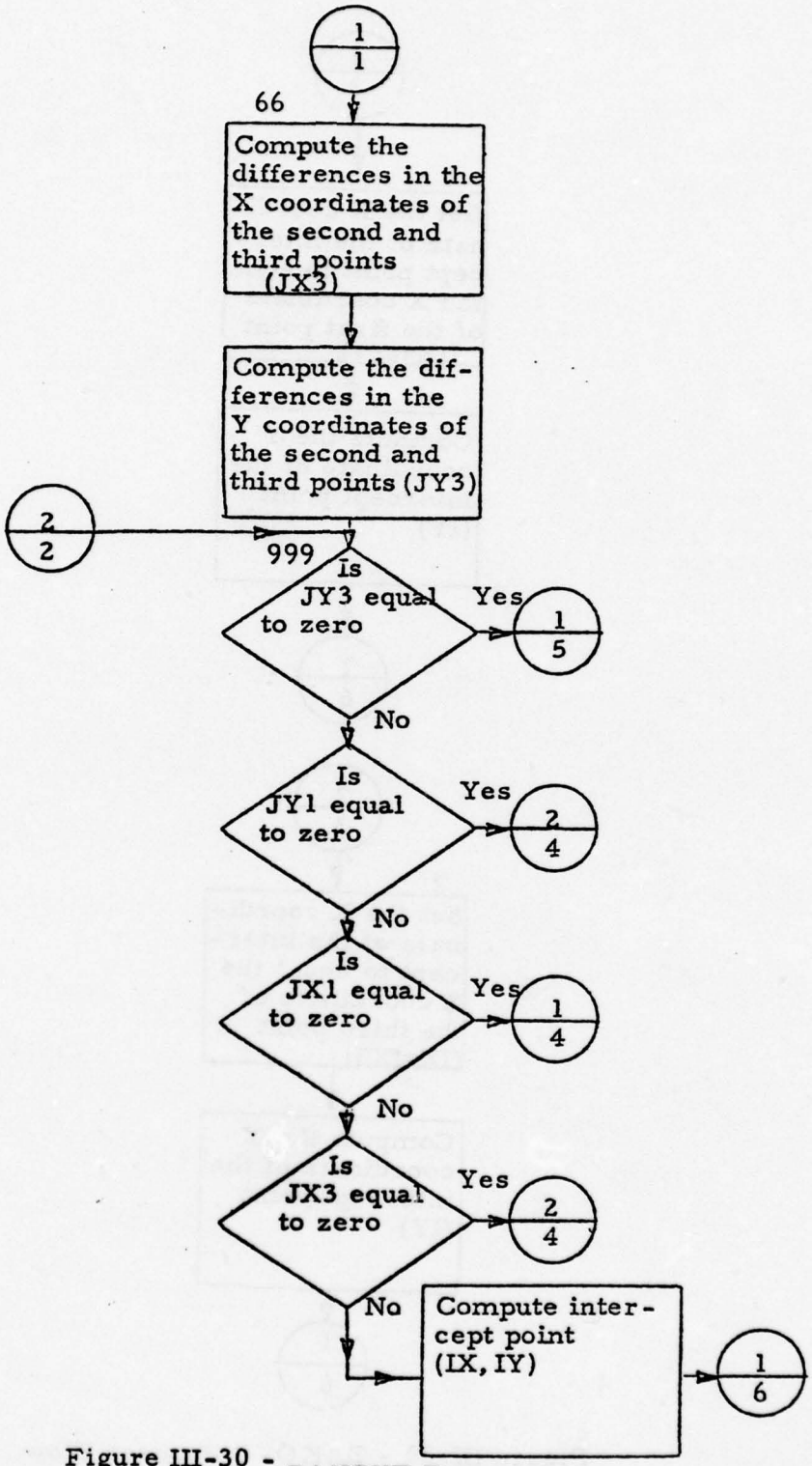



Figure III-30 - BAKOVE Process Flow (Page 3 of 6)

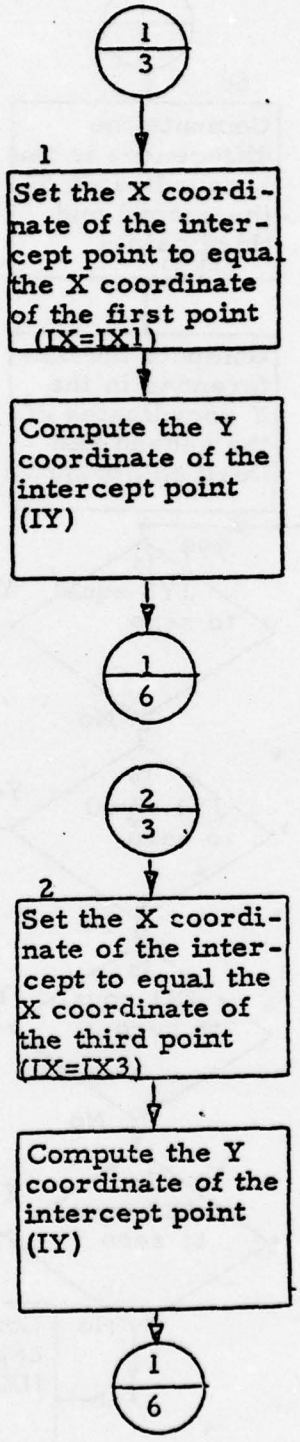



Figure III-30 - BAKOVE Process Flow
(Page 4 of 6)

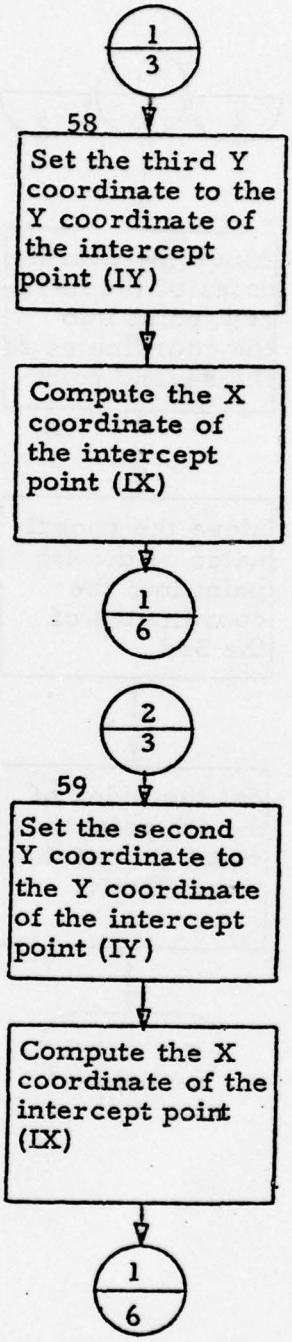



Figure III-30 - BAKOVE Process Flow
(Page 5 of 6)

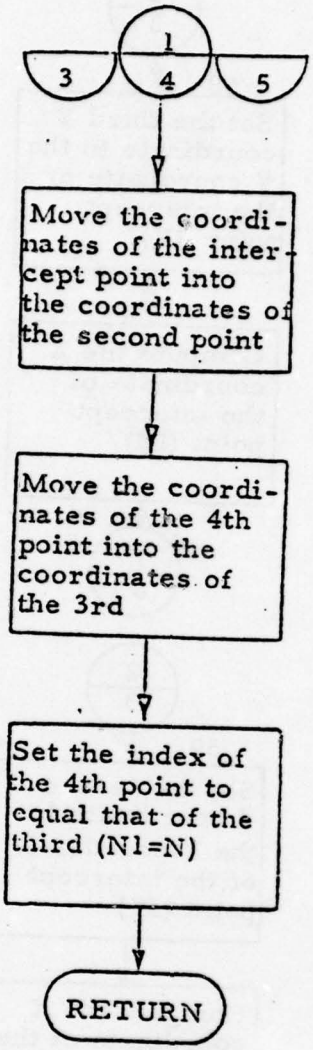



Figure III-30 - BAKOVE Process Flow

(Page 6 of 6)

## DD. CASEST

### 1. Functional Description

As a subroutine, CASEST computes two points on a perpendicular line to a line segment, through one of its endpoints.

### 2. Computer Definition

#### a. Core Memory Used

227 octal words.

#### b. Peripheral Equipment

None

### 3. Program Description

#### a. Calling Routines

CASER

#### b. Subroutines Used

None

#### c. Input

| | | |
|---|---|---|
| IX1, IY1 | - | coordinates of the first endpoint. |
| IX2, IY2 | - | coordinates of the second endpoint. |
| ID | - | distance between the endpoints. |
| IHC | - | distance on the perpendicular. |
| M | - | pointer to endpoint to be offset; if M=1, for the first endpoint, and if M=2, for the second endpoint. |

#### d. Output

| | | |
|---|---|---|
| IXL, IYL | - | coordinates of the left justified point on the perpendicular. |
| IXR, IYR | - | coordinates of the right justified point on the perpendicular. |

e. Processing Methodology

As a FORTRAN subroutine, CASEST, when called, will first compute the perpendicular line to the line segment defined by the two points ( (IX1, IY1), (IX2, IY2) ) supplied by the calling routine. A test is then made to determine at which point, the first or the second end-point, the perpendicular should be passed through by investigating the input variable M for 1 or 2. At the determined point, CASEST then computes the coordinates (IXL, IYL) of the left justified point and the coordinates (IXR, IYR) of the right justified point to the line segment at a distance of IHC. After computing the left and right justified points, CASEST returns control to the calling routine. See Figure III-31 for the processing flow diagram.

f. Calling Sequence

Call CASEST (M, IX1, IX2, IY1, IY2, ID, IXL, IYL, IXR, IYR, IHC).

g. Major Algorithms

The algorithm used in CASEST is the same as described in CASPOT (Sec. III-GG), except that only one pair of coordinate points are generated at either point one or two.

4. Program Constants and Variables

None

5. Error Conditions

None

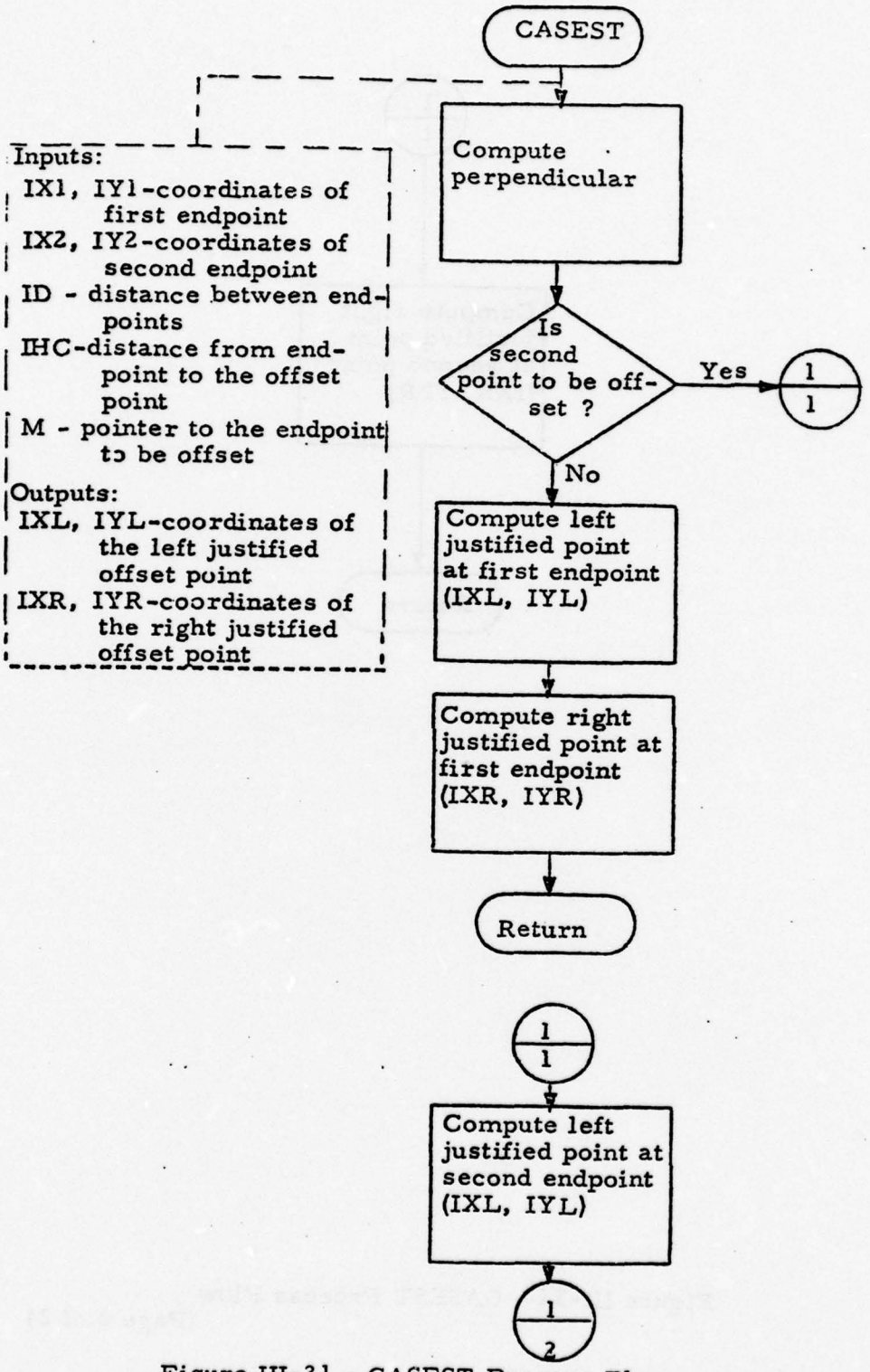



Figure III-31 - CASEST Process Flow

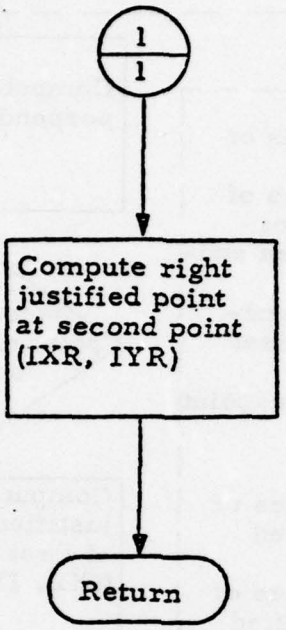



Figure III-31 - CASEST Process Flow
(Page 2 of 2)

## EE. CASEIT

### 1. Functional Description

As a subroutine, CASEIT computes two opposite points on a perpendicular bisect of a line segment defined by the endpoints.

### 2. Computer Definition

#### a. Core Memory Used

150 octal words.

#### b. Peripheral Equipment

None

### 3. Program Description

#### a. Calling Routines

CASER

#### b. Subroutines Used

None

#### c. Input

| | | |
|---|---|---|
| IX1, IY1 | - | coordinates of the first endpoint. |
| IX2, IY2 | - | coordinates of the second endpoint. |
| ID | - | distance (squared) between the endpoints. |
| IHC | - | distance on the perpendicular bisect from the line segment to each of the computed points. |

#### d. Output

| | | |
|---|---|---|
| IXL, IYL | - | coordinates of the left justified point on the perpendicular bisect. |
| IXR, IYR | - | coordinates of the right justified point on the perpendicular bisect. |

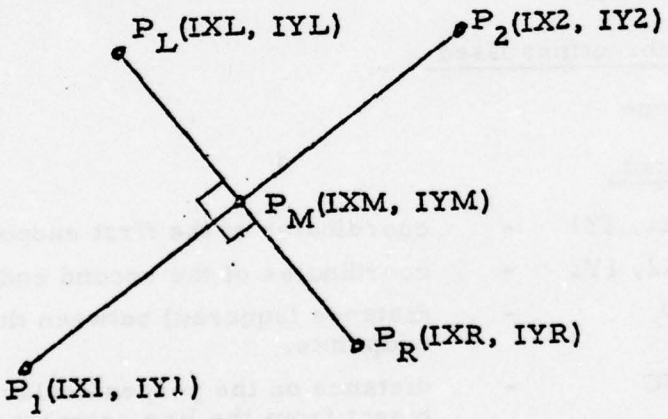
e. Processing Methodology

As a FORTRAN subroutine, CASEIT, when called, will first compute the coordinates (IXM, IYM) of the midpoint between the endpoints ( (IX1, IY1), (IX2, IY2) ) supplied by the calling routine. After computing the midpoint, CASEIT calculates the perpendicular bisect from which the coordinates of the left (IXL, IYL) and the right (IXR, IYR) justified points are determined at a distance of IHC from the midpoint. CASEIT now returns control to the calling routine. See Figure III-32 for the processing flow diagram.

f. Calling Sequence

Call CASEIT (IX1, IY1, IX2, IY2, ID, IXL, IYL, IXR, IYR, IHC).

g. Major Algorithms




ID - distance (squared) from point $P_1$ to point $P_2$.

IHC - distance from point $P_L$ to point $P_M$, also distance from point $P_R$ to point $P_M$.

$$IXM = IX1+IX2/2$$
$$IYM = IY1+IY2/2$$
$$IXL = IXM - (IY2-IY1) \cdot (IHC / ID)$$
$$IYL = IYM - (IX1-IX2) \cdot (IHC / ID)$$
$$IXR = IXM + (IY2-IY1) \cdot (IHC / ID)$$
$$IYR = IYM + (IX1-IX2) \cdot (IHC / ID)$$

4. Program Constants and Variables

IXM, IYM - coordinates for the midpoint.

5. Error Conditions

None

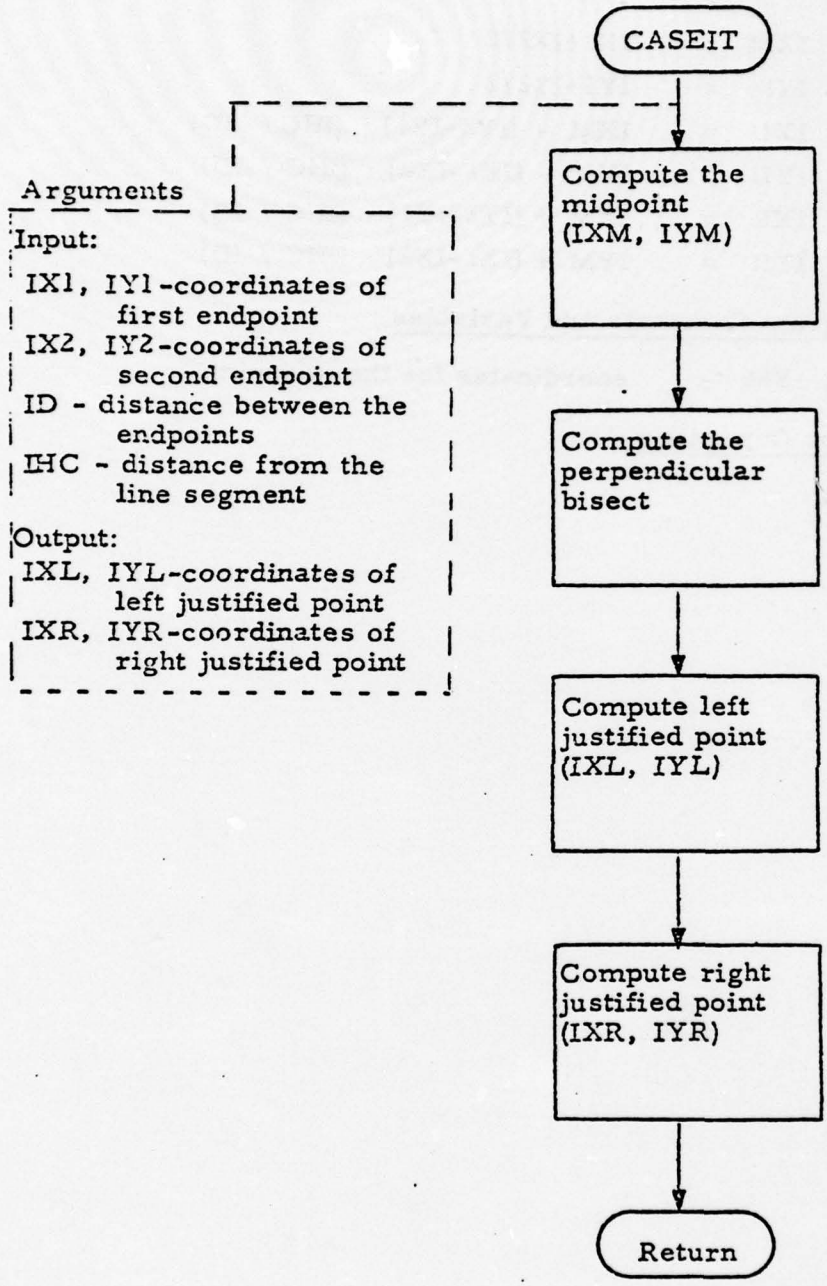



Figure III-32 - CASEIT Process Flow

(Page 1 of 1)

FF. CPDIST

1. Functional Description

As a subroutine, CPDIST deletes the second point of two points if the points are at a distance less than the specified minimum distance.

2. Computer Definition

a. Core Memory Used

44 octal words.

b. Peripheral Equipment

None

3. Program Description

a. Calling Routines

CASER

b. Subroutines Used

None

c. Input

| | | |
|---|---|---|
| IX1, IY1 | - | coordinates of first point. |
| IX2, IY2 | - | coordinates of second point. |
| N | - | index of the second point. |
| MINRES | - | minimum resolution (squared). |

d. Output

| | | |
|---|---|---|
| N | - | reduce index of the second point to equal the index at the first point. |

e. Processing Methodology

As a FORTRAN subroutine, CPDIST, when called, will first compute the distance (squared) between the two points which are

received from the calling routine. After computing the distance between the two points, CPDIST tests this distance against a minimum distance (MINRES). If the computed distance is less than the minimum distance, the index of the second point is decreased by one to suppress the second point. See Figure III-33 for the flow diagram.

f. Calling Sequence

Call CPDIST (IX1, IY1, IX2, IY2, N, MINRES).

g. Major Algorithms

None

4. Program Constants and Variables

None

5. Error Conditions

None

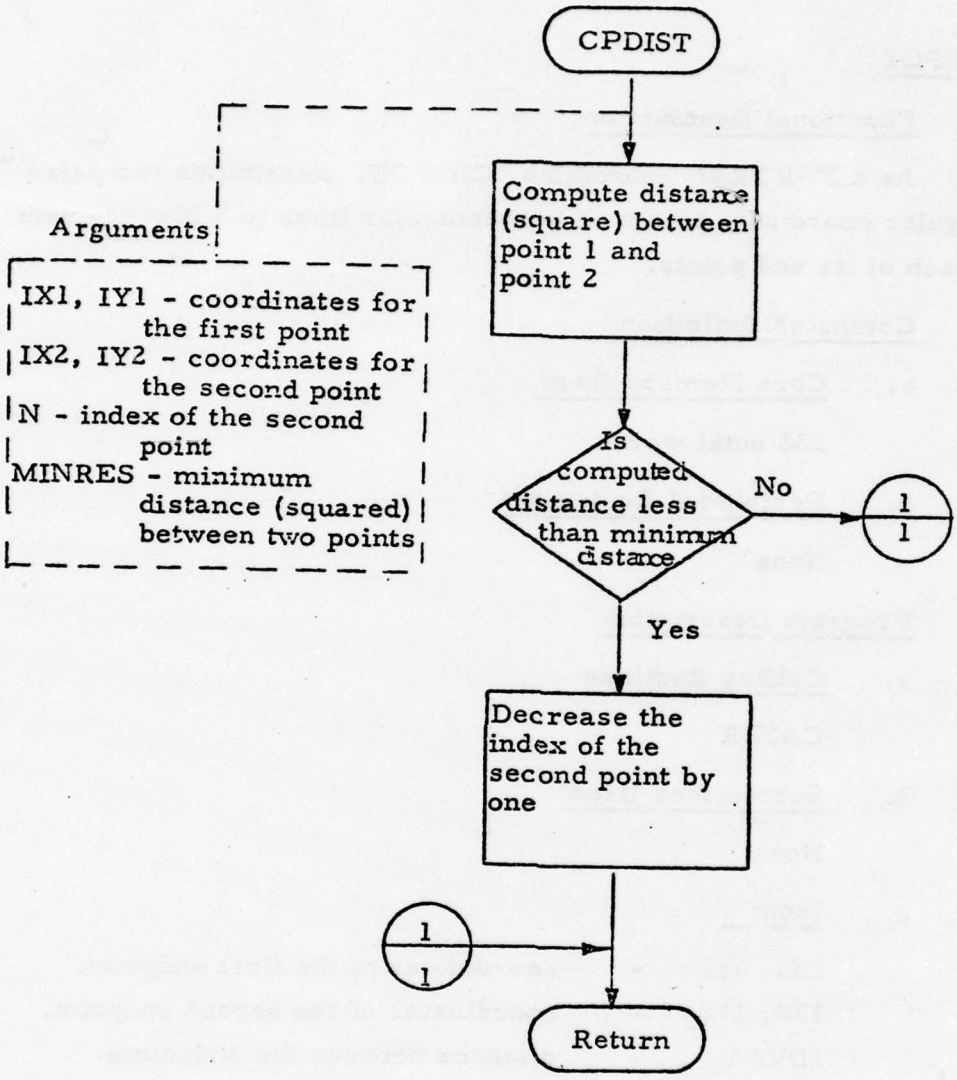



Figure III-33 - CPDIST Process Flow
(Page 1 of 1)

## GG. CASPOT

### 1. Functional Description

As a FORTRAN subroutine, CASPOT, determines two pairs of rectangular coordinate points on perpendicular lines to a line segment through each of its end points.

### 2. Computer Definition

#### a. Core Memory Used

232 octal words

#### b. Peripheral Equipment

None

### 3. Program Description

#### a. Calling Routines

CASER

#### b. Subroutines Used

None

#### c. Input

| | | |
|---|---|---|
| IX1, IY1 | - | coordinates of the first endpoint. |
| IX2, IY2 | - | coordinates of the second endpoint. |
| IDVAL | - | distance between the endpoints. |
| IHC | - | distance from endpoint to computed point. |

#### d. Output

| | | |
|---|---|---|
| IXL1, IYL1 | - | coordinates of the left justified point from the first endpoint. |
| IXR1, IYR1 | - | coordinates of the right justified point from the first endpoint. |

IXL2, IYL2 - coordinates of the left justified point from the second endpoint.

IXR2, IYR2 - coordinates of the right justified point from the second endpoint.

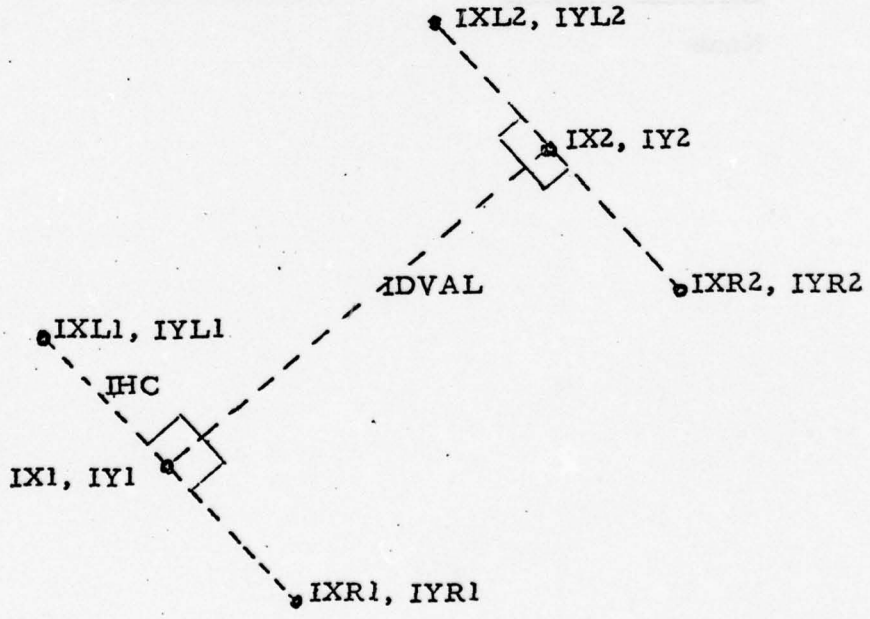
e. Processing Methodology

As a FORTRAN subroutine, CASPOT, when called, will first compute the perpendicular line to the line segment defined by the two endpoints ( (IX1, IY1), (IX2, IY2) ) supplied by the calling routine. Next CASPOT proceeds to compute the left and right justified points at each of the endpoints. After computing the left and right justified points, CASPOT returns control to the calling routine. See Figure III-34 for the processing flow diagram.

f. Calling Sequence

Call CASPOT (IX1, IY1, IX2, IY2, IDVAL, IXL1, IYL1, IXR1, IYR1, IXL2, IYL2, IXR2, IYR2, IHC).

g. Major Algorithms



```
X    =    (IX1-IX2) IHC/IDVAL
Y    =    (IY2-IY1) IHC/IDVAL
IXL1 =    IX1-Y
IYL1 =    IY1-X
IXR1 =    IX1+Y
IYR1 =    IY1+X
IXL2 =    IX2-Y
IYL2 =    IY2-X
IXR2 =    IX2+Y
IYR2 =    IY2+X
```

4. Program Constants and Variables

X - perpendicular distance in X direction.

Y - perpendicular distance in Y direction.

5. Error Conditions

None

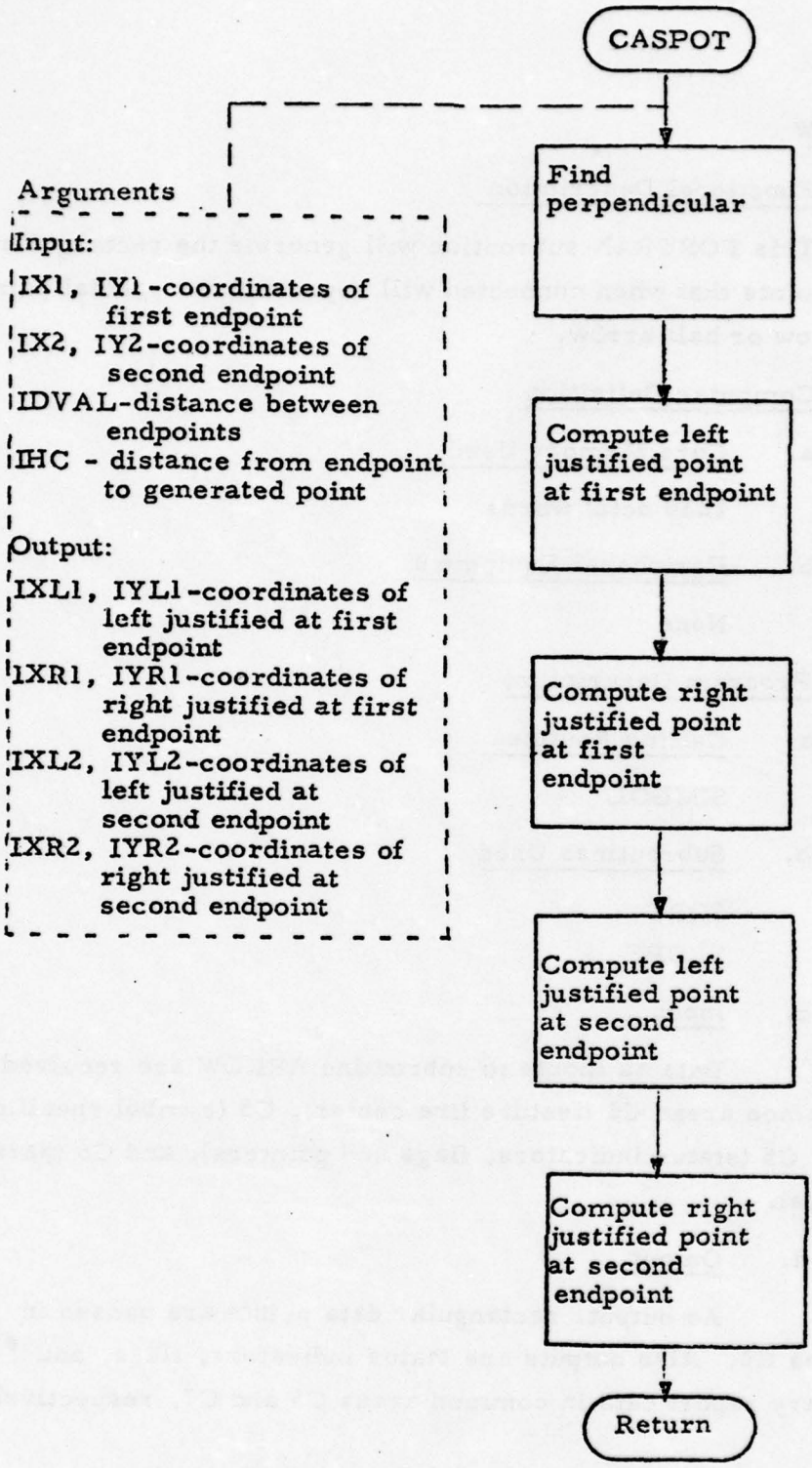



Figure III-34 - CASPOT Process Flow (Page 1 of 1)

## HH. ARROW

### 1. Functional Description

This FORTRAN subroutine will generate the rectangular coordinate points that when connected will represent the special point symbols arrow or half arrow.

### 2. Computer Definition

#### a. Core Memory Used

1230 octal words

#### b. Peripheral Equipment

None

### 3. Program Description

#### a. Calling Routines

SIMBOL

#### b. Subroutines Used

SQRT
SLOPE

#### c. Input

Data as inputs to subroutine ARROW are received through common areas C2 (feature line center), C3 (symbol specification directives), C5 (status indicators, flags and pointers), and C6 (parameters and variables).

#### d. Output

As output, rectangular data points are passed in common area C2. Also outputs are status indicators, flags, and tally summary report data in common areas C5 and C7, respectively.

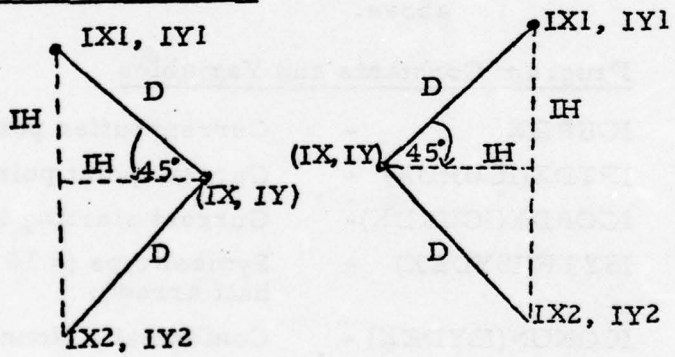
e. Processing Methodology

When called, the FORTRAN subroutine ARROW will first locate the center point at which the special point symbol of an arrow or half arrow is to be generated and set the appropriate flags for output. Using the size (ISYSZ (ISYDEX) ) and whether confomal or nonconformal (ICONON (ISYDEX) = 0 or 1), ARROW will compute the difference along the x axis and y axis where the generated points will be located from the center point. ARROW proceeds now to determine if an arrow or half arrow is to be generated by interrogating ISYTP (ISYDEX). Next the coordinate values are computed and entered into the IXYZ buffer in the common area C2. After the indicator and flags for output are set, control is returned to the calling routine. Refer to Figure III-35 for the process flow diagram of ARROW.

f. Calling Sequence

CALL ARROW

g. Major Algorithms




| | | |
|---|---|---|
| D | - | length of arrow |
| IX, IY | - | center point at which arrow is to be generated |
| IH | - | $D \cdot \sin 45^\circ = D \cdot (0.70711)$ |

| | | |
|---|---|---|
| S | - | interpreted slope |
| M1 | - | $(IH/\sqrt{1 + (S \cdot S)}) \cdot (1-S)$ |
| M2 | - | $(IH/\sqrt{1 + (S \cdot S)}) \cdot (1+S)$ |

| At beginning of line | At end of line |
|---|---|
| Nonconformal: | |
| IX1 = IX - IH | IX1 = IX + IH |
| IY1 = IY + IH | IY1 = IY + IH |
| IX2 = IX - IH | IX2 = IX + IH |
| IY2 = IY - IH | IY2 = IY - IH |
| Conformal: | |
| IX1 = IX - M2 | IX1 = IX + M1 |
| IY1 = IY - M1 | IY1 = IY + M2 |
| IX2 = IX + M1 | IX2 = IX + M2 |
| IY2 = IY - M2 | IY2 = IY - M1 |

Note: Half arrow requires only one point from the above.

4. Program Constants and Variables

| | | |
|---|---|---|
| ICURDX | - | Current buffer pointer. |
| IPTDX(ICURDX) | - | Current point pointer. |
| ICORDX(ICURDX) | - | Current starting index pointer. |
| ISYTP(ISYDEX) | - | Symbol type (= 10 arrow; = 11 half arrow). |
| ICONON(ISYDEX) | - | Conformal - Nonconformal data. |
| ISYSZ(ISYDEX) | - | Symbol size. |
| IXYZ(I, J) | - | X, Y value along with distance. |
| IX, IY | - | Center pointer. |
| 0.70711 | - | Sin 45°. |
| NUMPTS(I) | - | Number of points in buffer I. |
| ICURDX | - | Current buffer pointer. |

| | | |
|---|---|---|
| ISYRDY | - | Symbol ready for output flag. |
| IH | - | Distance along X and Y axis for nonconformal arrow. |
| M1 and M2 | - | Distance along axis for conformal arrow. |
| S | - | Interrogated slope. |

5. Error Conditions

None

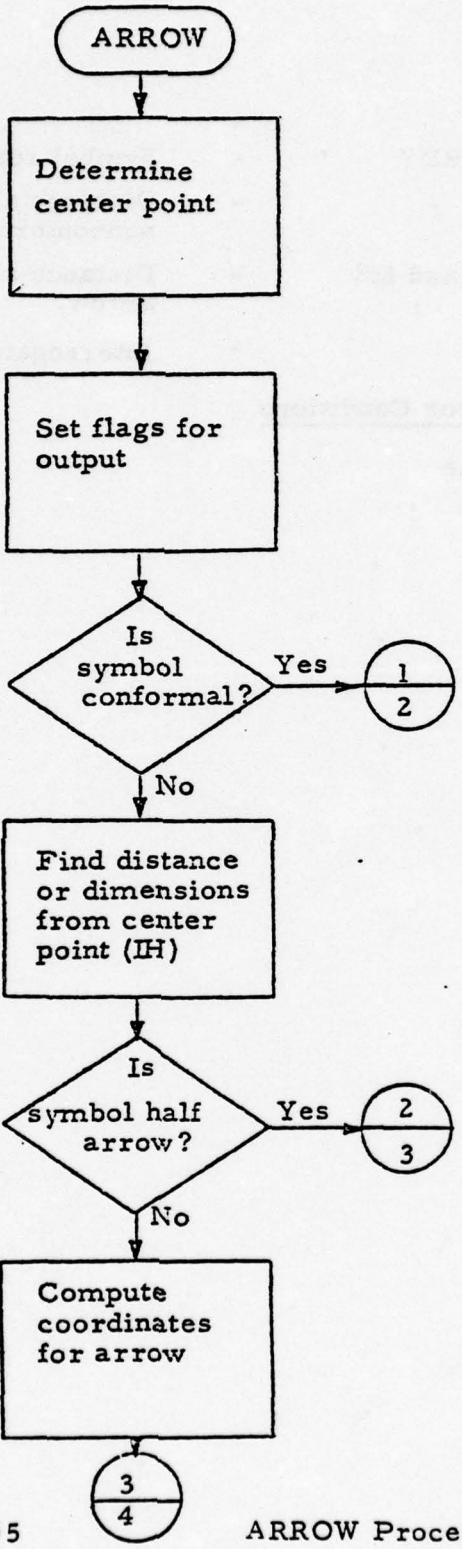



Figure III-35 ARROW Process Flow

(Page 1 of 4)

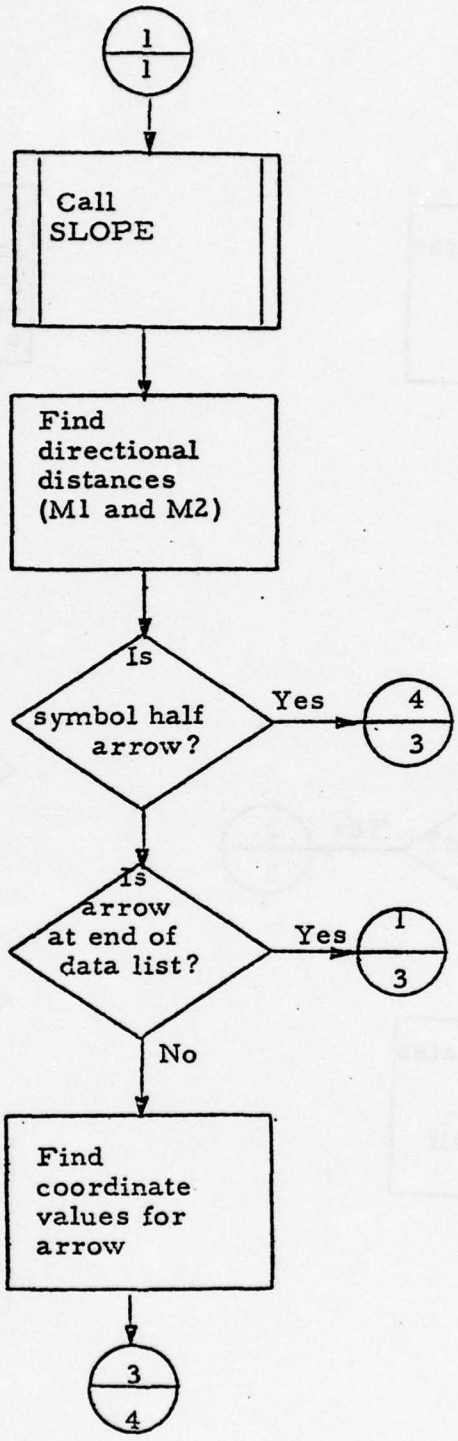



Figure III-35 - ARROW Process Flow
(Page 2 of 4)

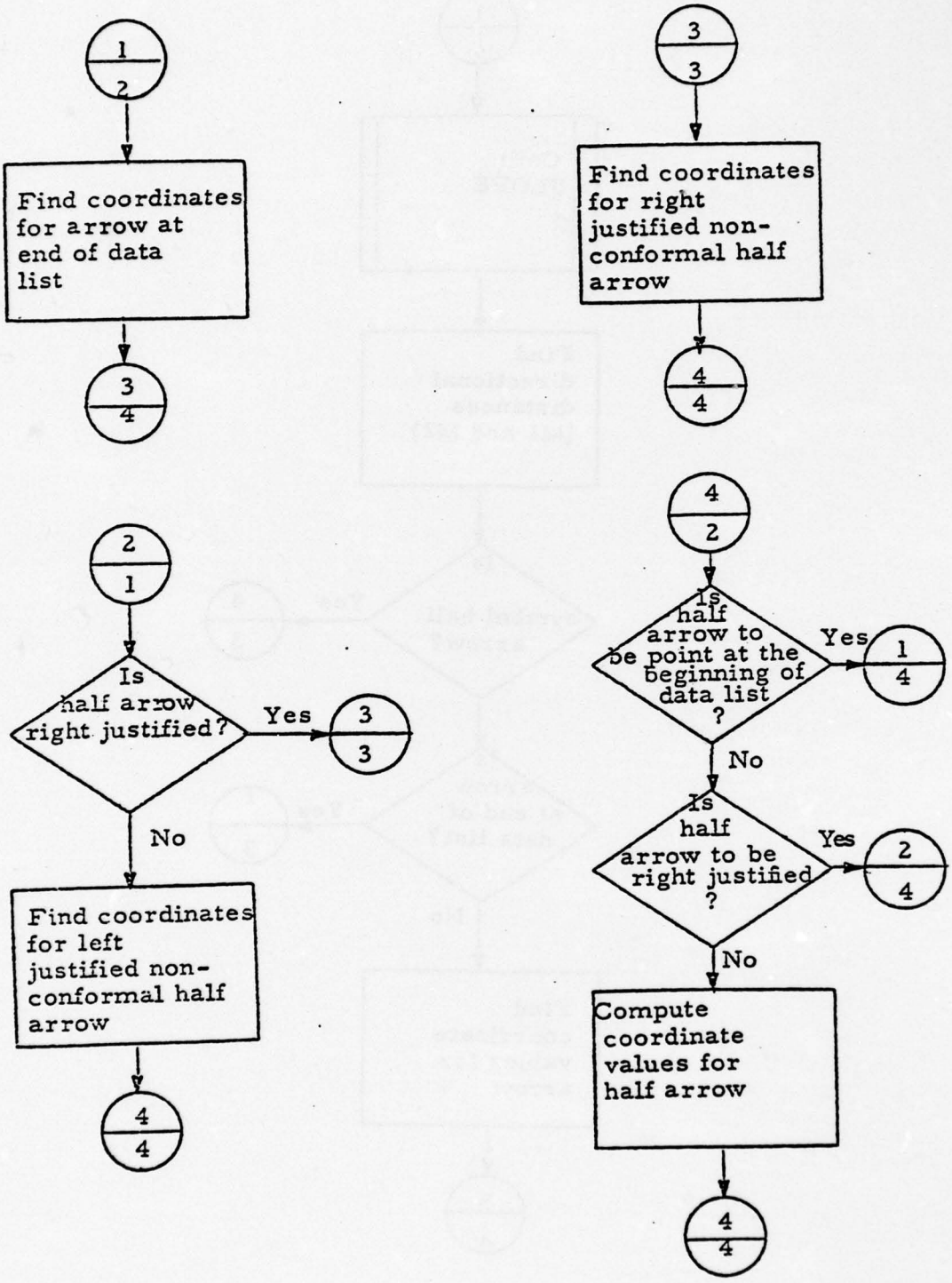



Figure III-35 - ARROW Process Flow

(Page 3 of 4)

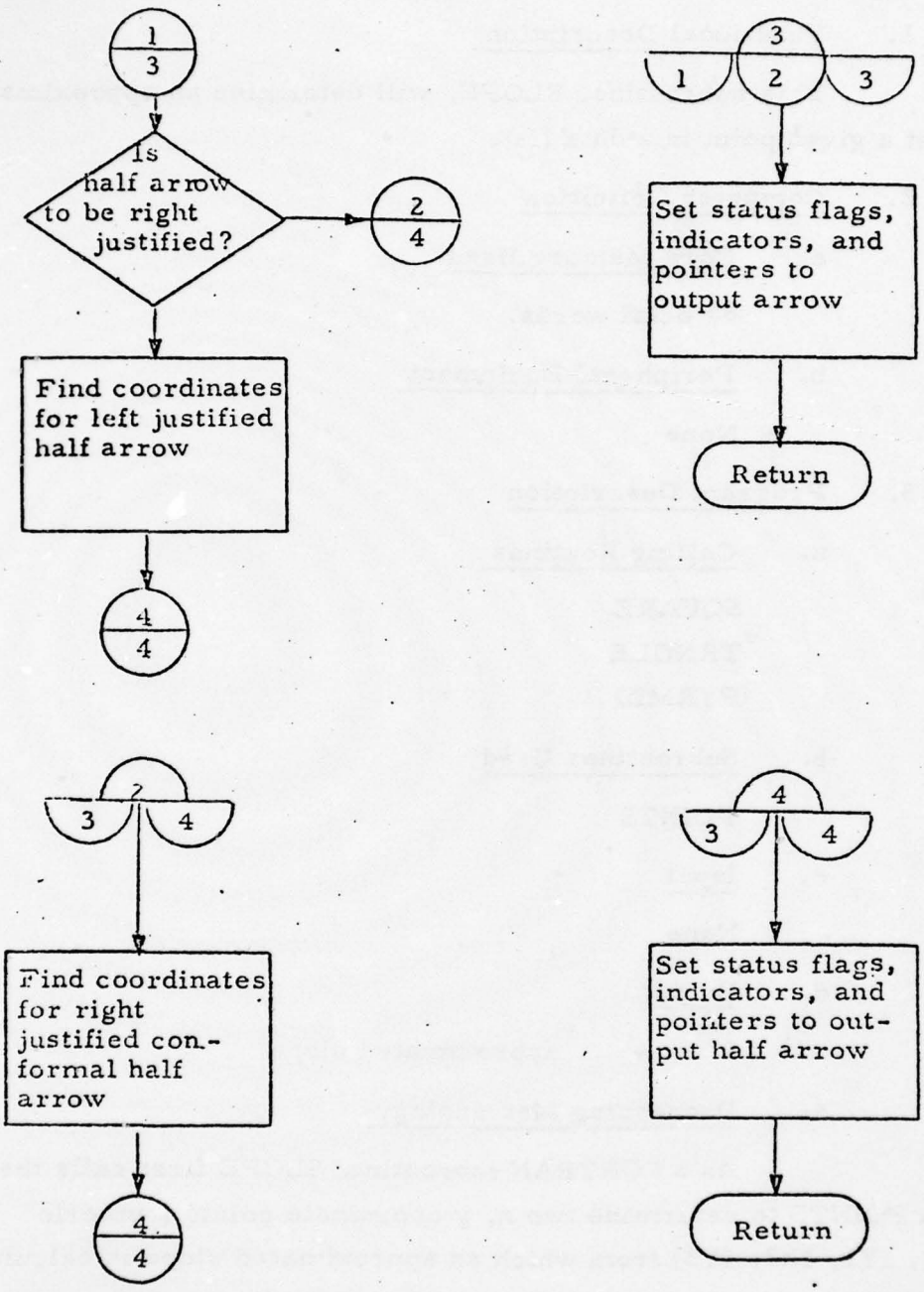



Figure III-35 - ARROW Process Flow
(Page 4 of 4)

II. SLOPE

1. Functional Description

This subroutine, SLOPE, will determine an approximate slope about a given point in a data list.

2. Computer Definition

a. Core Memory Used

63 octal words.

b. Peripheral Equipment

None

3. Program Description

a. Calling Routines

SQUARE
TRNGLE
PYRMID

b. Subroutines Used

POINTS

c. Input

None

d. Output

S - approximated slope

e. Processing Methodology

As a FORTRAN subroutine, SLOPE first calls the subroutine POINTS to determine two x, y coordinate points (numeric names IX1, IY1, IX2, IY2) from which an approximated slope is calculated (see the description of POINTS). From the two coordinate points

generated by POINTS, SLOPE finds the difference in the x coordinates and tests this difference against zero (i.e. test if slope is paralleled with the y axis). If the difference in the x coordinate is zero, SLOPE sets the slope(s) equal to 1.0E + 31 and control is returned to the calling routine. If the difference in the x coordinates is not zero, SLOPE proceeds in computing the approximate slope (numeric name S) by determining the difference of the y coordinates over the difference of the x coordinate. See Figure III-36 for processing flow diagram.

f. Calling Sequence

Call SLOPE (S)

g. Major Algorithms

None

4. Program Constants and Variables

None

5. Error Conditions

None

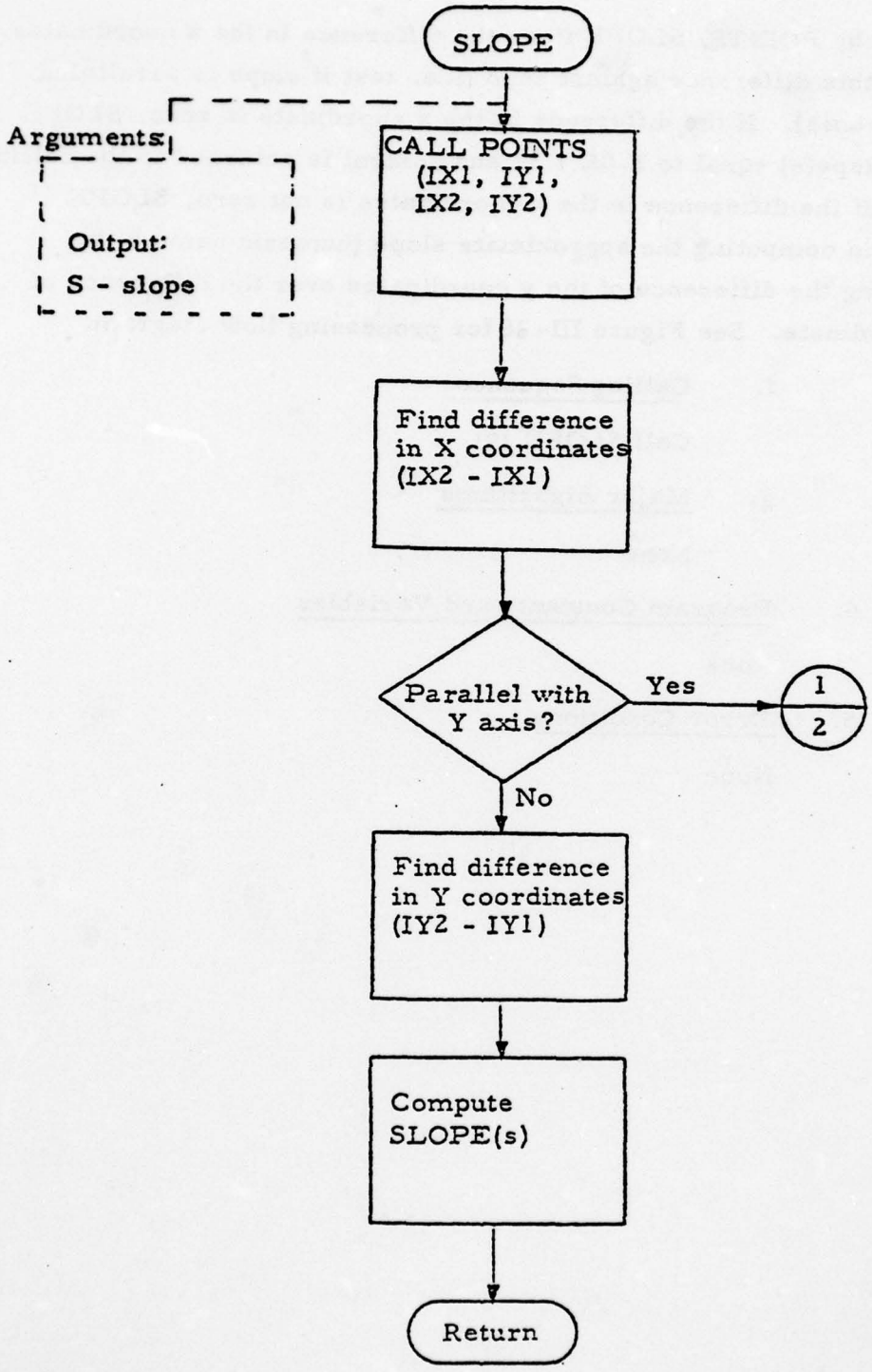



Figure III-36 SLOPE Process Flow (Page 1 of 2)

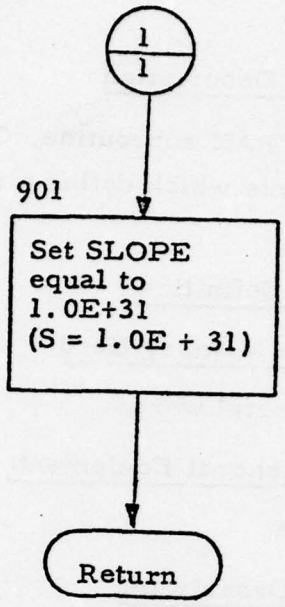



Figure III-36 SLOPE Process Flow (Page 2 of 2)

## JJ. CROSS

### 1. Functional Description

As a FORTRAN subroutine, CROSS computes two pairs of rectangualr coordinate points which define a cross.

### 2. Computer Definition

#### a. Core Memory Used

417 octal words.

#### b. Peripheral Equipment

None

### 3. Program Description

#### a. Calling Routines

SIMBOL

#### b. Subroutines Used

SQRT
SLOPE

#### c. Input

Inputs to subroutine CROSS consist of data contained in common areas C2 (feature line center data), C3 (symbol specifications directives), C5 (status indicators flags and pointers), and C6 (parameters and variables).

#### d. Output

Output will consist of four coordinate data points in common area C2, and also consist of status indicator flags and tally summary report data in common area C5 and C7, respectively.

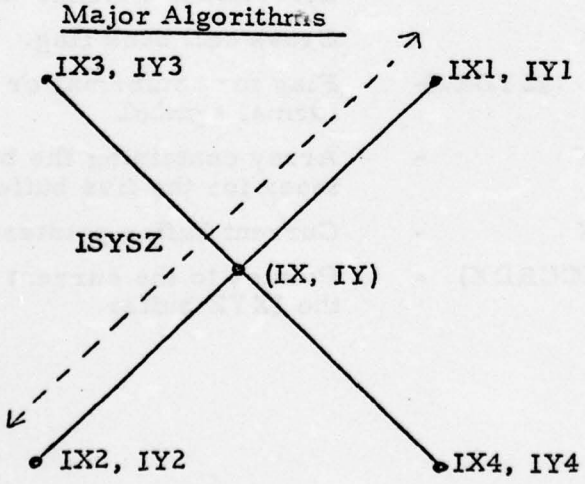
e. Processing Methodology

When called upon, the subroutine CROSS will first check to see if its internal flag JCALL has been set. If JCALL has been set, CROSS will proceed to set and/or clear the appropriate status flags and indicators to output the second line of the cross. If JCALL is off (= 0), CROSS will proceed to determine the center point from the IXYZ buffer (common area C2) about which the cross will be computed. Then CROSS determines the size of the cross and whether the cross is to be conformal or nonconformal by interrogating ISYSZ and ICONON respectively (contain in common area labeled C3). Then ALPHA and BETA is determined (see algorithms). Using ALPHA and BETA, CROSS computes the four coordinate values for the cross while placing one pair in the third section of buffer IXYZ and the second pair in the fourth section of buffer IXYZ. CROSS now sets the appropriate status flags and indicators to output one side (first pair of coordinate values) and sets the flags in common area C5 such that control will be returned to CROSS. Control is now returned to the calling routine. See Figure III-37 for the processing flow diagram.

f. Calling Sequence

Call CROSS

g. Major Algorithms



ISYSZ - Size of cross.

IX, IY - Center point about which the cross is to be drawn.

S - Slope.

Nonconformal:

IALPHA = 1/2 (0.70711 · ISYSZ)

IBETA = 1/2 (0.70711 · ISYZ)

Conformal:

$$IALPHA = 1/2\ ISYSZ \cdot 0.70711 \cdot (1.0+S^2)^{-1/2}(1.0-S)$$

$$IBETA = 1/2\ ISYSZ \cdot 0.70711 \cdot (1.0+S^2)^{-1/2}(1.0+S)$$

IX1 = IX + IALPHA

IY1 = IY + IBETA

IX2 = IX - IALPHA

IY2 = IY - IBETA

IX3 = IX - IBETA

IY3 = IY + IALPHA

IX4 = IX + IBETA

IY4 = IY - IALPHA

4. Program Constants and Variables

IALPHA - Directional distance in X direction.

IBETA - Directional distance in Y direction.

ICLLBK - Cross call back flag.

ICONCON(ISYDEX) - Flag for conformal or nonconformal symbol

ICORDX - Array containing the beginning index for the five buffers in IXYZ.

ICURDX - Current buffer pointer.

IPTDX(ICURDX) - Pointer to the current point in the IXYZ buffer.

| | | |
|---|---|---|
| ISPDST | - | Distance to approximate slope. |
| ISYDEX | - | Index into the symbol spec directives. |
| ISYRDY | - | Symbol piece ready for output flag. |
| ISYTP(ISYDEX) | - | Size of symbol. |
| ITELRN | - | At the end of data points flag. |
| IX | - | X coordinate for center point. |
| IY | - | Y coordinate for center point. |
| IXYZ | - | Two-dimensional array containing X, Y coordinate points with distance between two successive points. |
| JCALL | - | Internal flag denoting whether to output second side or generate the cross. |
| NUMCBK | - | Number of call back flags. |
| NUMCRS | - | Number of crosses generated. |
| NUMPTS | - | Number of points per buffer. |
| S | - | Approximate slope. |

5. Error Conditions

None

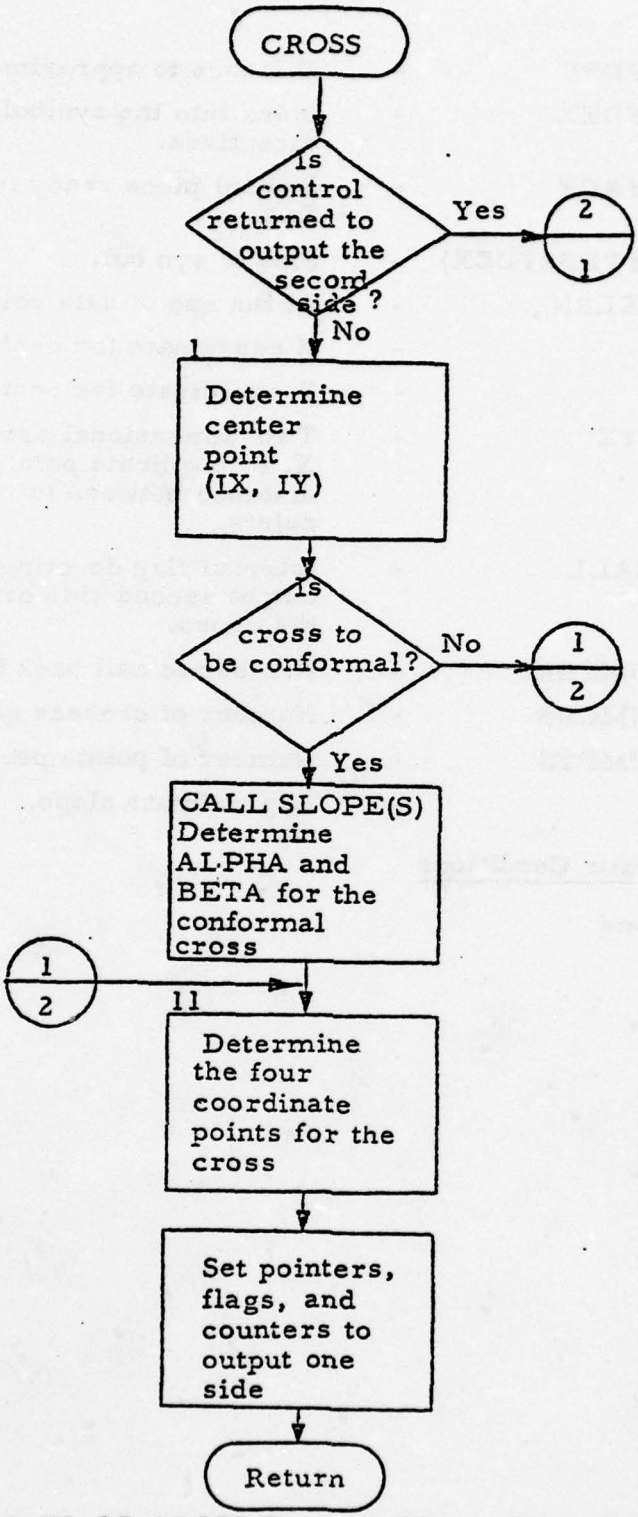



Figure III-37 CROSS Process Flow (Page 1 of 2)

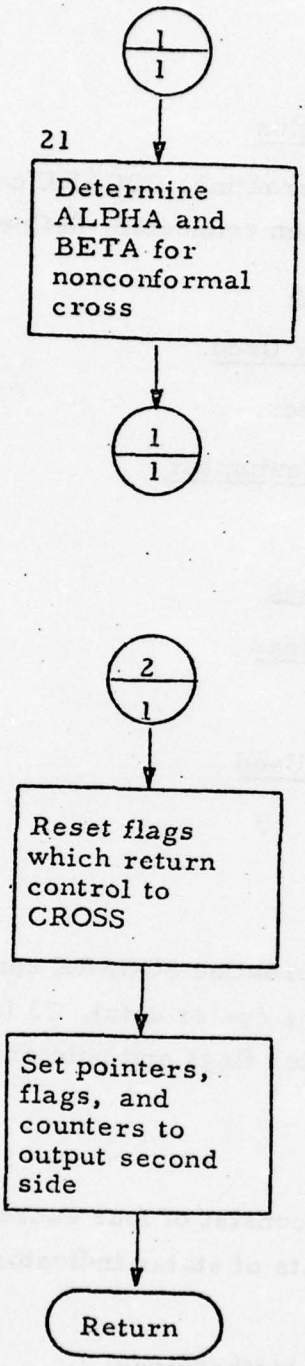



Figure III-37 - CROSS Process Flow (Page 2 of 2)

KK. SQUARE

1. Functional Description

As a FORTRAN subroutine, SQUARE computes four rectangular coordinate points, such that when connected, defines a square.

2. Computer Definition

a. Core Memory Used

314 octal words.

b. Peripheral Equipment

None.

3. Program Description

a. Calling Routines

SIMBOL

b. Subroutines Used

SLOPE
SQRT

c. Input

Inputs to subroutine SQUARE consist of data contained in common areas C2 (feature line center data), C3 (symbol specifications directives), C5 (status indicator flags and pointers), and C6 (parameters and variables.

d. Output

Output will consist of four coordinate data points in common area C2, and also consists of status indicator flags in common area C5.

e. Processing Methodology

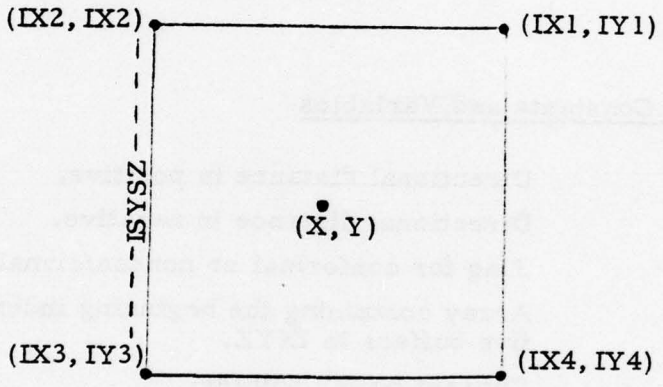
When called upon, the subroutine SQUARE will first determine the center point in the IXYZ buffer about which the coordinate points

for the square will be computed. Then SQUARE determines the size of the square and whether the square is to be conformal or nonconformal by interrogating ISYSZ and ICONON, respectively (contained in common area C3). Then the slope (S), CHI, and CHI1 are determined (see Algorithms). Using CHI and CHI1, SQUARE computes the four coordinate points which are stored in the third section of buffer IXYZ. SQUARE now sets the appropriate status flags and indicators to output the coordinate points for the symbol piece square.

f. Calling Sequence

CALL SQUARE

g. Major Algorithms




| | |
|---|---|
| ISYSZ | Size of square |
| (X, Y) | Center point about which the square is to be generated. |
| S | slope, for a nonconformal square slope equals zero. |

For conformal square:

$$CHI = \tfrac{1}{2} \cdot ISYSZ \cdot (1.0+5) \cdot (1.0+5 \cdot 5)^{-\frac{1}{2}}$$

$$CHI1 = \tfrac{1}{2} \cdot ISYSZ \cdot (1.0-5) \cdot (1.0+5 \cdot 5)^{-\frac{1}{2}}$$

For nonconformal square:

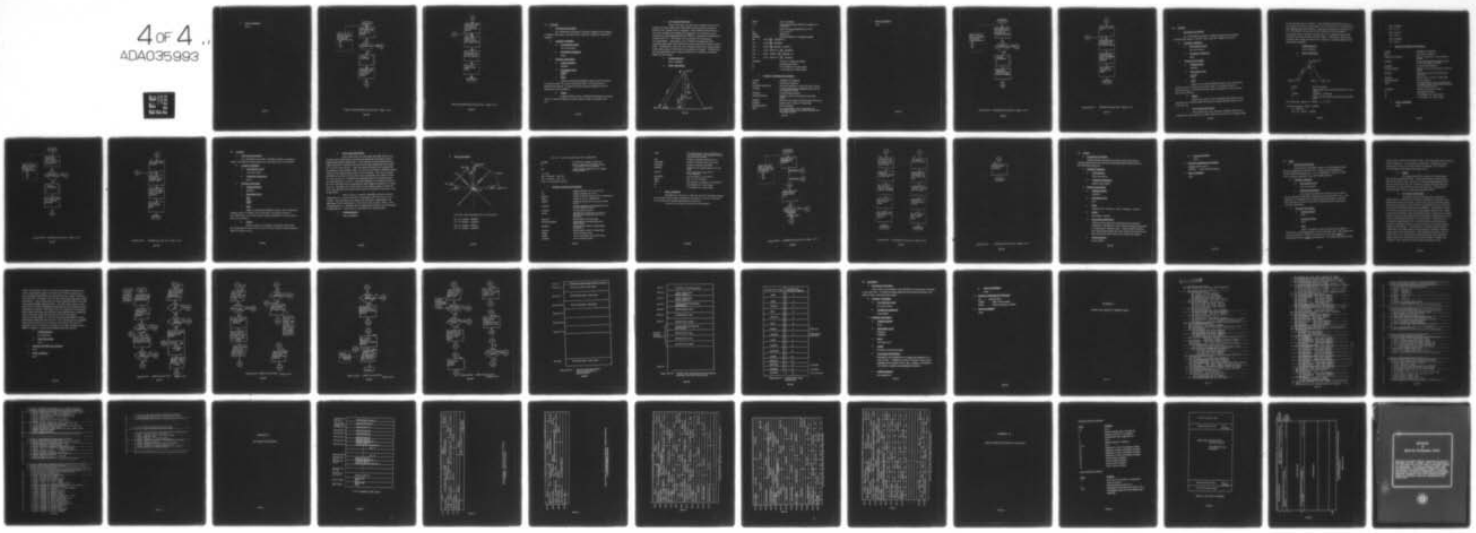
```
CHI = ISYSZ/2
CHI1 = CH1

IX1 = X + CHI1
IY1 = Y + CH1

IX2 = X-CHI
IY2 = Y + CHI1

IX3 = X - CHI1
IY3 = Y- CHI

IX4 = X + CHI
IY4 = Y-CHI1
```

4. Program Constants and Variables

| | |
|---|---|
| CHI | Directional distance in positive. |
| CHI1 | Directional distance in negative. |
| ICONCON-(ISYDEX) | Flag for conformal or nonconformal symbol. |
| ICONDX | Array containing the beginning index for the five buffers in IXYZ. |
| ICURDX | Current buffer pointer. |
| IPTDX(ICURDX) | Pointer to the current point in the IXYZ buffer. |
| ISYDEX | Index into the symbol specifications directives. |
| ISYRDY | Symbol piece ready for output flag. |
| ISYSZ(ISYDEX) | Size of symbol. |
| IXYZ | Two dimensional array containing X, Y coordinate points with distance between two successive points. |
| NUMPTS | Number of points per buffer. |
| S | Approximate slope. |
| X | X coordinate for center point. |
| Y | Y coordinate for center point. |

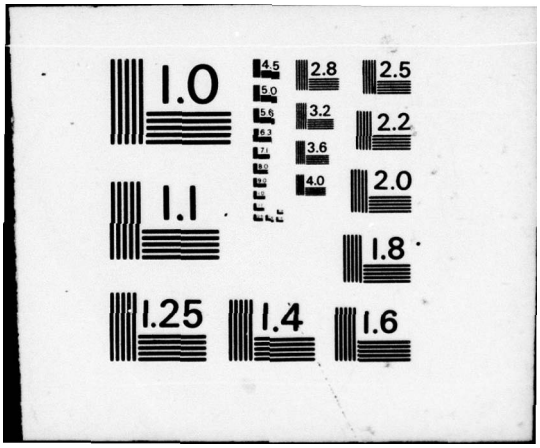
AD-A035 993 PRC INFORMATION SCIENCES CO MCLEAN VA F/G 8/2
ACS SYMBOLIZATION FOR DMAAC. VOLUME II. COMPUTER PROGRAM DOCUME--ETC(U)
NOV 76 P D BELL, J A NEUFFER, M L TAYLOR F30602-75-C-0319
UNCLASSIFIED RADC-TR-76-334-VOL-2 NL

4 OF 4
ADA035993



END

DATE
FILMED
3-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

5. **Error Conditions**

None.

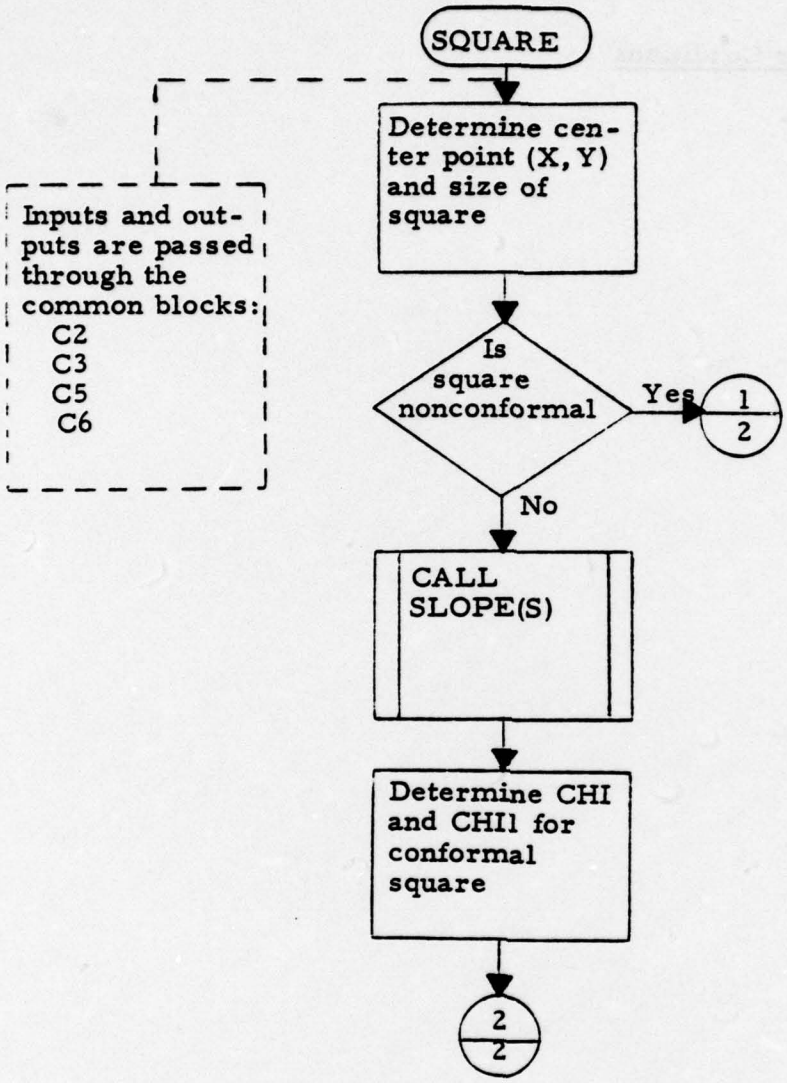



Figure III-38 -SQUARE Process Flow (Page 1 of 2)

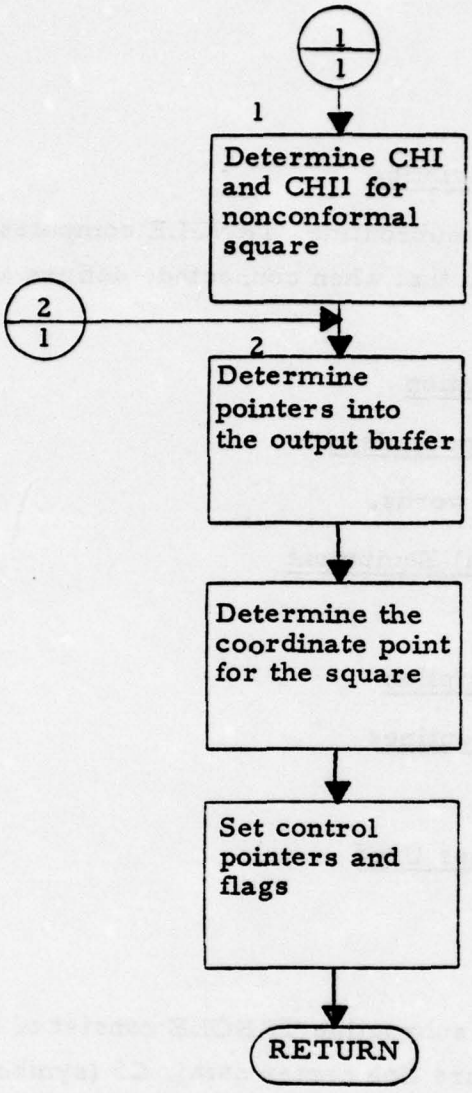



Figure III-38 SQUARE Process Flow (Page 2 of 2)

## LL. TRNGLE

### 1. Functional Description

As a FORTRAN subroutine, TRNGLE computes three distinct coordinate data points, such that when connected, defines an equilateral triangle.

### 2. Computer Definition

#### a. Core Memory Used

326 octal words.

#### b. Peripheral Equipment

None.

### 3. Program Description

#### a. Calling Routines

SIMBOL

#### b. Subroutines Used

SLOPE
SQRT

#### c. Input

Inputs to subroutine TRNGLE consist of data contained in common areas C2 (feature line center data), C3 (symbol specifications directives), C5 (status indicator flags and pointers), and C6 (parameters and variables).

#### d. Output

Output consists of four coordinate data points in common area C2, and also consists of status indicator flags in common area C5.

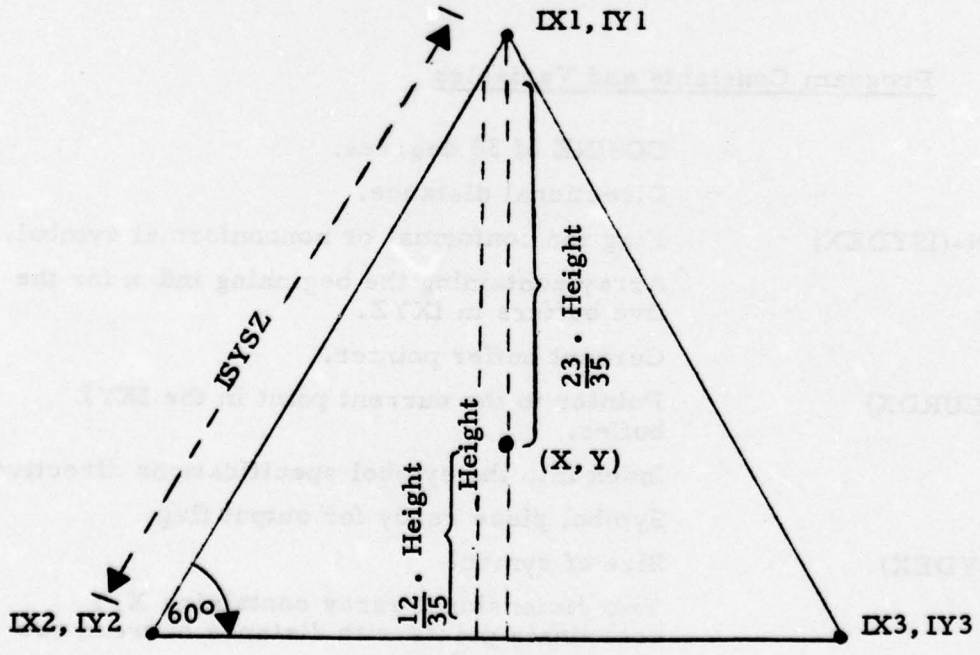
e. Processing Methodology

When called upon, the subroutine TRNGLE will first determine the center point in the IXYZ buffer about which the coordinate points for the triangle will be computed. Then TRNGLE determines the size of the triangle and whether the triangle is to be conformal or non-conformal by interrogating ISYSZ and ICONON, respectively (contained in common area C3). Then BETA and the slope (S) are determined (see algorithms). Using BETA and S, TRNGLE computes the three coordinate data points while placing the points in the third section of buffer IXYZ. TRNGLE also stores the first computed point in the fourth location of the third section of IXYZ buffer to complete the triangle. TRNGLE now sets the appropriate status flags and indicators to output the coordinate points for the triangle.

f. Calling Sequence

CALL TRNGLE

g. Major Algorithms



| | |
|---|---|
| ISYSZ | Size of triangle |
| X, Y | Center point about which the triangle is to be drawn. |
| S | Slope (for a nonconformal set S=0.0 |
| SL | $(1.0 + S \cdot S)^{-\frac{1}{2}}$ |
| BETA | ISYSZ/2.0 |
| HEIGHT | COS $(30^{\circ})$*ISYSZ = 0.8660255. ISYSZ |

IX1 = $X - SL \cdot (\frac{23}{35} \cdot HEIGHT \cdot S)$

IY1 = $Y + SL \cdot (\frac{23}{35} \cdot HEIGHT)$

IX2 = $X + SL \cdot (\frac{12}{35} \cdot HEIGHT \cdot S - BETA)$

IY2 = $Y - SL \cdot (BETA \cdot S + \frac{12}{35} \cdot HEIGHT)$

IX3 = $X + SL \cdot (BETA + \frac{12}{35} \cdot HEIGHT \cdot S)$

IY3 = $Y + SL \cdot (BETA \cdot S - \frac{12}{35} \cdot HEIGHT)$

| | |
|---|---|
| NUMPTS | Number of points per buffer. |
| S | Approximate slope. |
| X | X coordinate for center point. |
| Y | Y coordinate for center point |

4. Program Constants and Variables

| | |
|---|---|
| ALPHA | COSINE of 30 degrees. |
| BETA | Directional distance. |
| ICONCON-(ISYDEX) | Flag for conformal or nonconformal symbol. |
| ICONDX | Array containing the beginning index for the five buffers in IXYZ. |
| ICURDX | Current buffer pointer. |
| IPTDX(ICURDX) | Pointer to the current point in the IXYZ buffer. |
| ISYDEX | Index into the symbol specifications directives. |
| ISYRDY | Symbol piece ready for output flag. |
| ISYSX(ISYDEX) | Size of symbol. |
| IXYZ | Two dimensional array containing X, Y coordinate points with distance between two successive points. |

5. **Error Conditions**

None.

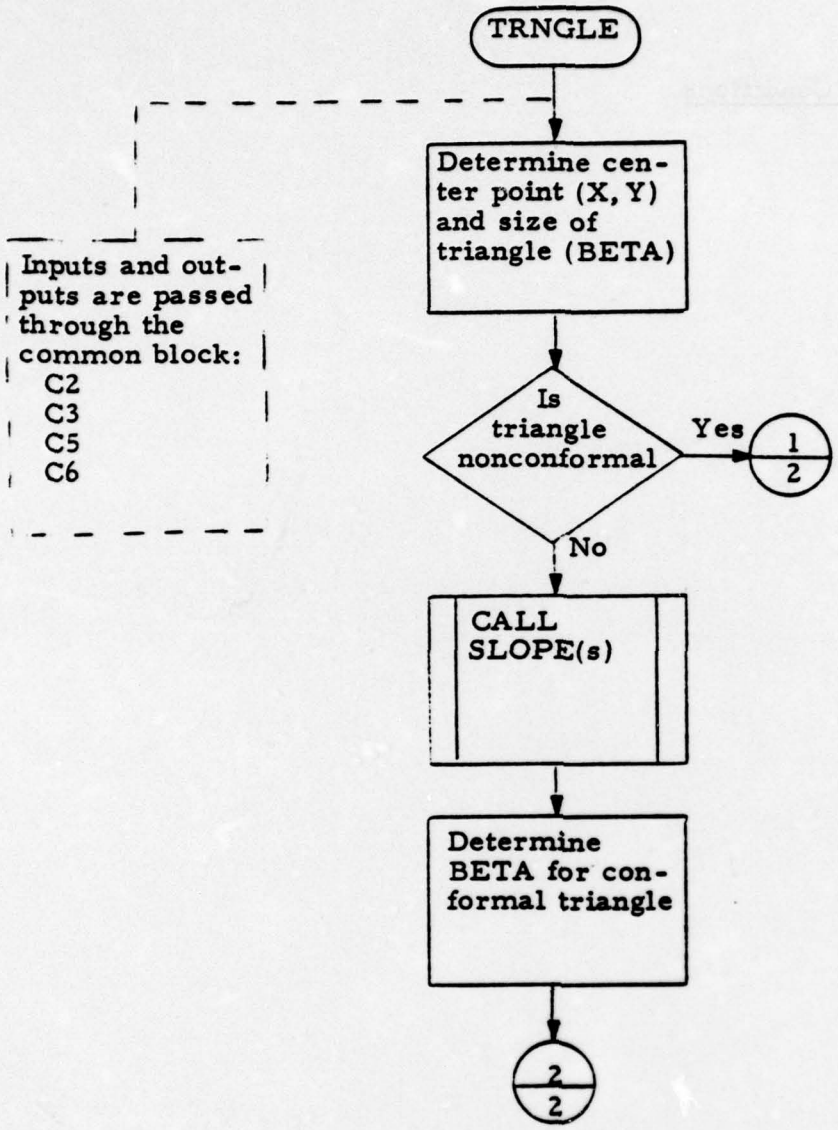



Figure III-39 - TRNGLE Process Flow (Page 1 of 2)

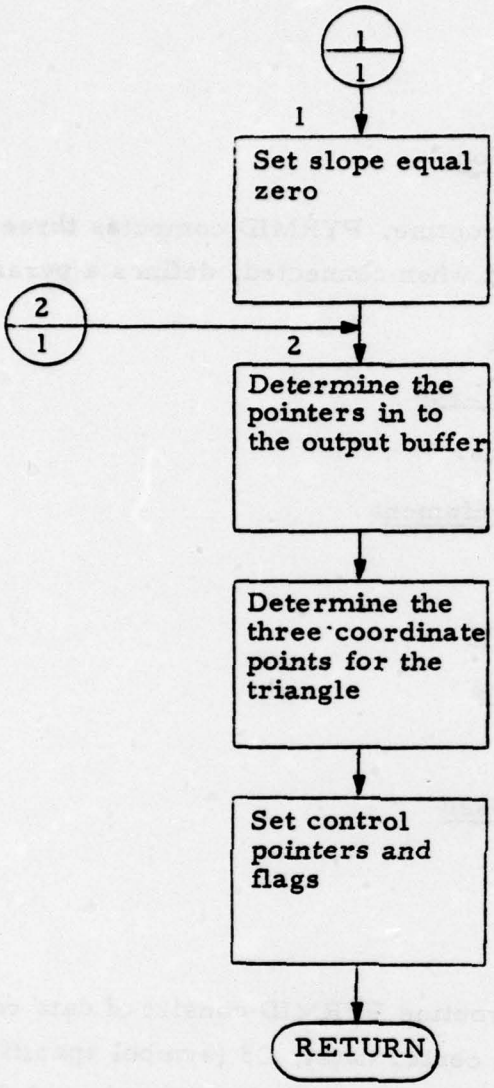



Figure III-39 - TRNGLE Process Flow (Page 2 of 2)

MM. PYRMID

1. Functional Description

As a FORTRAN subroutine, PYRMID computes three distinct coordinate data points, such that when connected, defines a pyramid.

2. Computer Definition

a. Core Memory Used

260 octal words.

b. Peripheral Equipment

None.

3. Program Description

a. Calling Routine

SIMBOL

b. Subroutines Used

SLOPE
SQRT

c. Input

Inputs to subroutine PYRMID consist of data contained in common areas C2 (feature line center data), C3 (symbol specifications directives), C5 (status indicator flags and pointers), and C6 (parameters and variables).

d. Output

Output will consist of four coordinate data points in common area C2, and also output will consist of status indicator flags in common area C5.

e. Processing Methodology

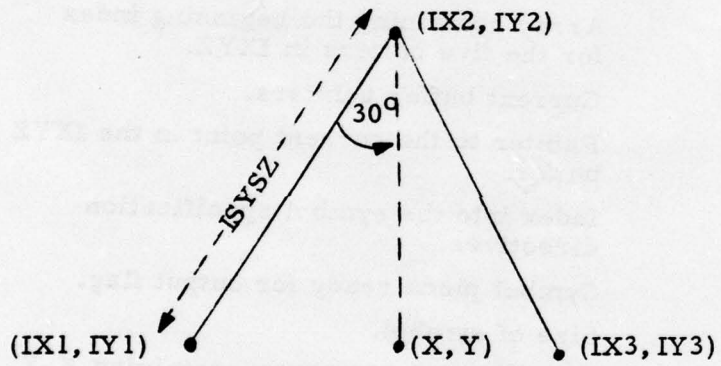
When called upon, the subroutine PYRMID will first determine the center point in the IXYZ buffer about which the coordinate points

for the pyramid will be computed. Then PYRMID determines the size of the pyramid and whether the pyramid is to be conformal or nonconformal by interrogating ISYSZ and ICONON, respectively (contained in common area C3). Then BETA and the slope (S) are determined (see algorithms). Using BETA and S, PYRMID computes the three coordinate points which are placed in the third section of buffer IXYZ. PYRMID now sets the appropriate status flags and indicators to output the coordinate points for the symbol piece pyramid.

f. Calling Sequence

CALL TRNGLE

g. Major Algorithms




| | |
|---|---|
| IXYXZ | Size of pyramid. |
| X, Y | Center point about which the pyramid is to be drawn. |
| ALPHA | $COS(30^{o}) = 0.8660255$ |
| S | Slope, for nonconformal pyramid slope equals zero. |

For Conformal, $BETA = \frac{1}{2} \cdot ISYSZ \cdot (1 + S \cdot S)^{-\frac{1}{2}}$

For Nonconformal, $BETA = \frac{1}{2} \cdot ISYSZ$

$C1 = BETA \cdot S$

$C2 = 2.0 \cdot BETA \cdot ALPHA$

IX1 = X-BETA

IY1 = Y-C1

IX2 = X-C2·S

IY2 = Y+C2

IX3 = X+BETA

IY3 = Y+C1

4. Program Constants and Variables

| | |
|---|---|
| ALPHA | COSINE of 30 degrees. |
| BETA | Directional distance. |
| ICONCON-(ISYDEX) | Flag for conformal or nonconformal symbol. |
| ICONDX | Array containing the beginning index for the five buffers in IXYZ. |
| ICURDX | Current buffer pointers. |
| IPTDX(ICURDX) | Pointer to the current point in the IXYZ buffer. |
| ISYDEX | Index into the symbol specification directives. |
| ISYRDY | Symbol piece ready for output flag. |
| ISYSZ(ISYDEX) | Size of symbol. |
| IXYZ | Two dimensional array containing X, Y coordinate points with distance between two successive points. |
| NUMPTS | Number of points per buffer. |
| S | Approximate slope. |
| X | X coordinate for center point. |
| Y | Y coordinate for center point. |

5. Error Conditions

None.

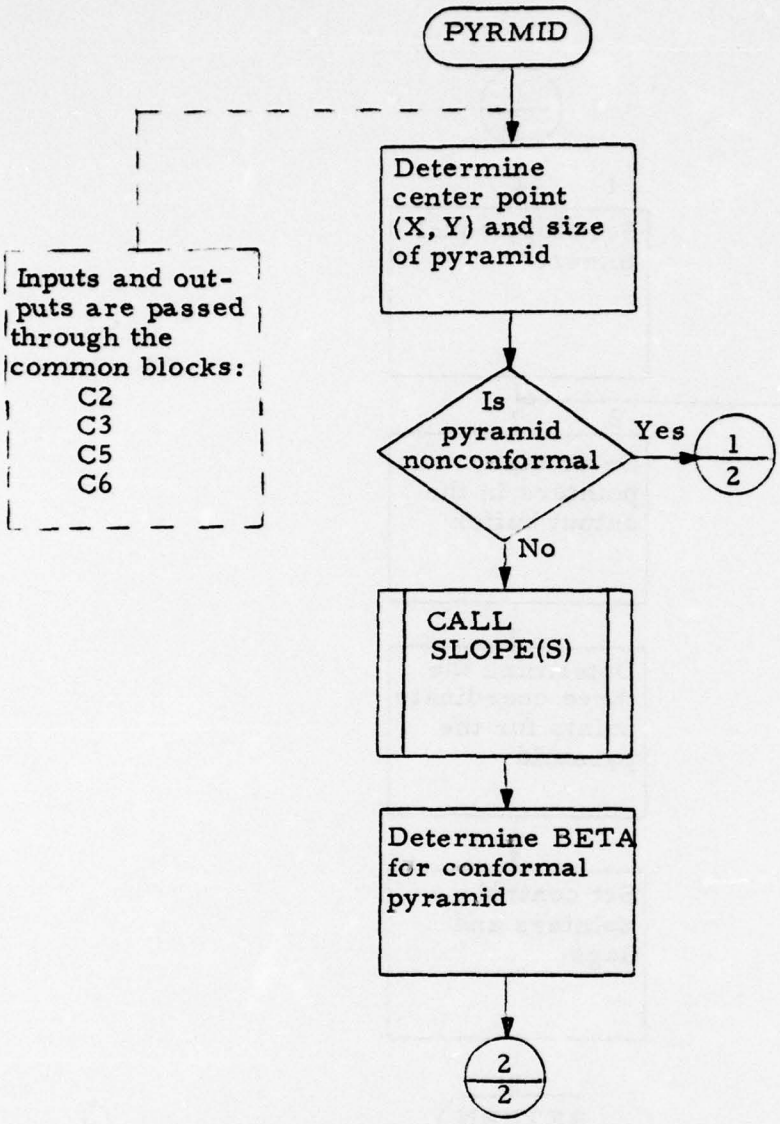



Figure III-40 - PYRMID Process Flow (Page 1 of 2)

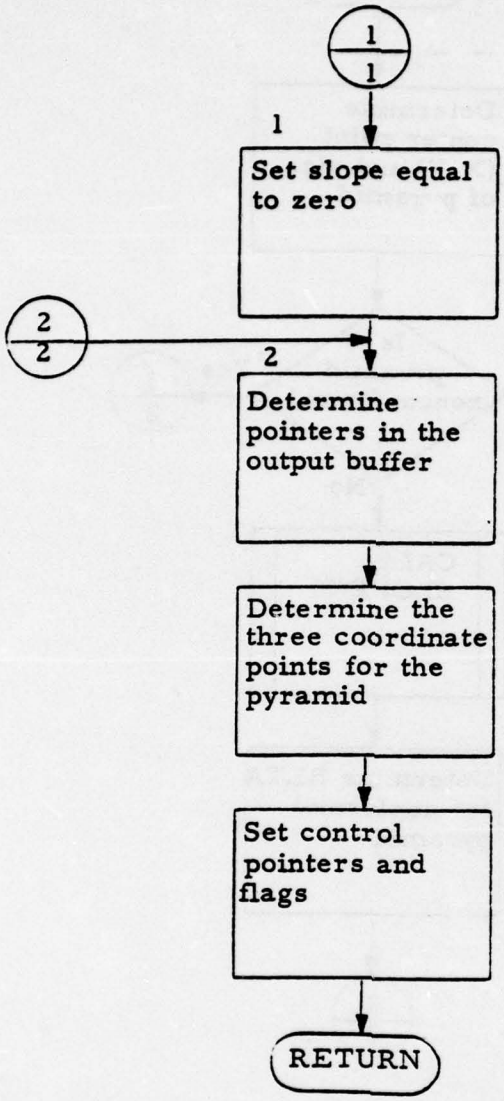



Figure III-40 - PYRMID Process Flow (Page 2 of 2)

NN. ARCORD

1. Functional Description

As a FORTRAN subroutine, ARCORD computes coordinate points, such that the symbol pieces of a cord and an arc are produced.

2. Computer Definition

a. Core Memory Used

511 octal words.

b. Peripheral Equipment

None.

3. Program Description

a. Calling Routines

SIMBOL.

b. Subroutines Used

SIN
COS
ATAN
SQRT

c. Input

Inputs to subroutine ARCORD consist of data contained in common areas C2 (feature line center data), C3 (symbol specifications directives), C5 (status indicator flags and pointers), and C6 (parameters and variables).

d. Output

Output consists of coordinate data points which define the symbol pieces of an arc and a cord, and also consist of status indicator flags in common area C5.

e. Processing Methodology

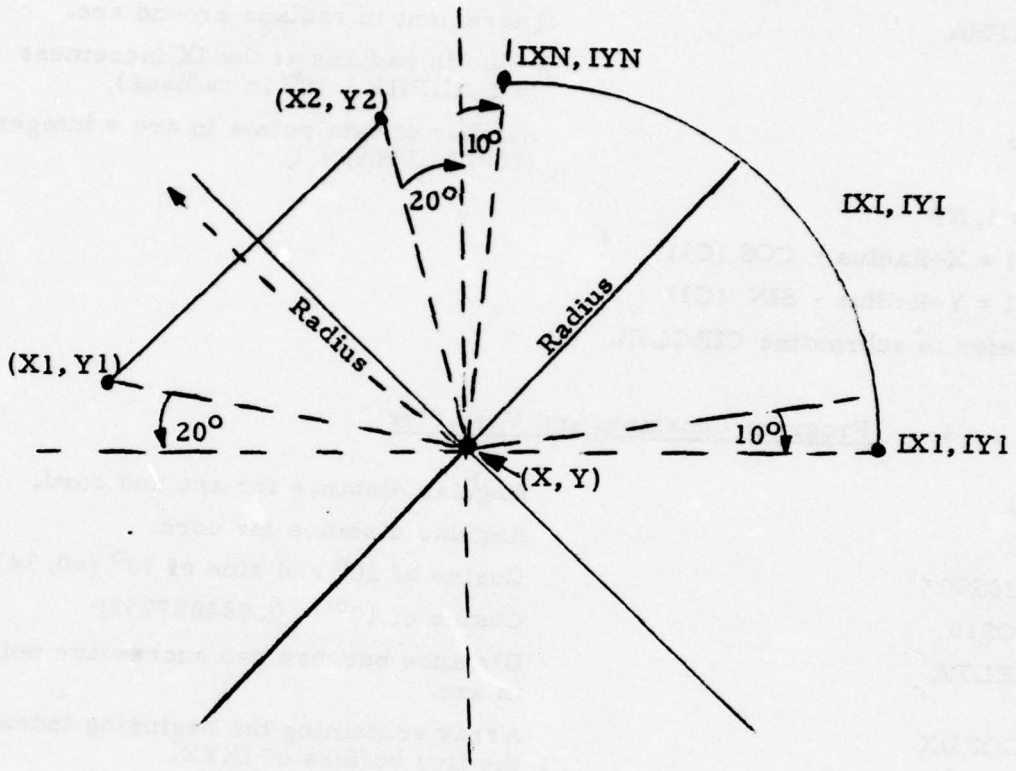
When called upon, the subroutine ARCORD will first determine if the symbol piece cord has been generated and output by interrogating the internal flag IFLAG. If the cord has not been generated and output (IFLAG=1), ARCORD will proceed with determining the center point (X,Y) for the cord and arc. Then the symbol directives are searched for the existence of the symbol cross. If a cross is not found, ARCORD returns control to the calling routine with an error message. If a cross is found, ARCORD uses the given size of the cross to determine the radius for both the cord and the arc. ARCORD will determine the two coordinate points (see algorithms) for the cord. After determining the coordinate points for the cord, ARCORD sets the appropriate status flags and indicators to output the cord. ARCORD will return control to the calling routine to output the cord, but ARCORD will request that control be returned so that the arc will be generated.

When control is returned, ARCORD proceeds with generating the arc. First the number of data points (NF) for the arc is calculated. Then ARCORD will compute the data points sequentially (see algorithms) while storing them in the third section of buffer IXYZ. After completing the arc, ARCORD sets the appropriate indicator and status flags to output the arc. Control now is returned to the calling routine.

f. Calling Sequence

CALL ARCORD

g. Major Algorithms




For cord - line from point (X1, Y1) to (IX2, IY2)

X1 = X - Radius · SIN(20°)
Y1 = Y + Radius · COS(20°)

X2 = X - Radius · SIN(70°)
Y2 = Y + Radius · COS(70°)

For arc - curve from point (IX1, IY1) to IXN, IYN)

| | |
|---|---|
| ALPHA | Increment in radians around arc. |
| C1 | Value in radians at the IX increment (= I·ALPHA + 10° in radians). |
| NP | number of data points in arc = integer (70°/ALPHA) + 1. |

I = 1, NP

IXI = X+Radius · COS (C1)

IYI = Y+Radius · SIN (C1)

(Refer to subroutine CIRCLE).

4. Program Constants and Variables

| | |
|---|---|
| C1 | Angular distance for arc and cord. |
| C2 | Angular distance for cord. |
| C20S70 | Cosine of 20° and sine of 70° (=0.342020143) |
| COS10 | Cosine of 10° (= 0.984807753) |
| DELTA | Distance between two successive points in arc. |
| ICORDX | Array containing the beginning index for the five buffers of IXYZ. |
| ICURDX | Current buffer pointer. |
| IFLAG | Internal flag to generate the cord (if IFLAG=1) or to generate the arc (if IFLAG=2). |
| IPECLK | Symbol piece call back flag. |
| IPTDX(ICURDX) | Index into the current point in the IXYZ buffer. |
| ISYDEX | Index into the symbol specifications directives. |
| ISYRDY | Symbol piece ready for output flag. |
| ISYSZ | Size of symbol array. |
| ISYTP | Array containing symbol piece type. |
| ITXERR | Error message buffer. |

| | |
|---|---|
| LXYZ | Two dimensional array containing X, Y coordinate points with distance between two successive points. |
| NP | Number of points contained in arc. |
| NUMCBK | Number of symbol piece call back. |
| NUMPTS | Number of points per buffer. |
| RADIUS | Radius for the symbol pieces arc and cord. |
| S20C70 | Sine of 20° and cosine of 70° (= 0.93969262). |
| SEVENTY | 70° in radians (=1.22173051). |
| SIN10 | Sine 10° (=0.1736481). |
| TEN | Ten degrees in radians (=0.174532930). |
| X | X coordinate for center point. |
| Y | Y coordinate for center point. |

5. Error Conditions

ARCORD will set the error flag and enter the following message into the error text location (ITXERR) if the symbol cross was not located or not found in the symbol specifications directives.

A cross was not found to associate with ARCORD.

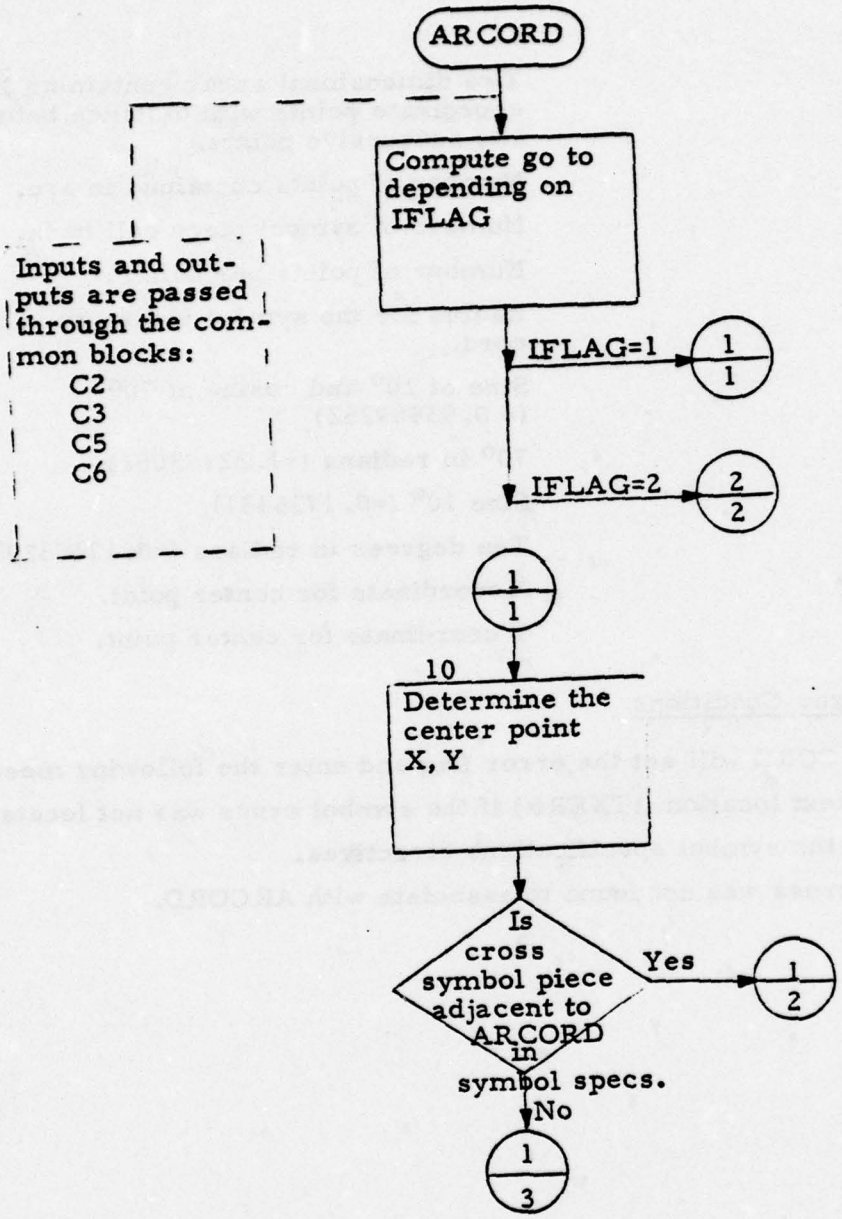



Figure III-41 - ARCORD Process Flow (Page 1 of 3)

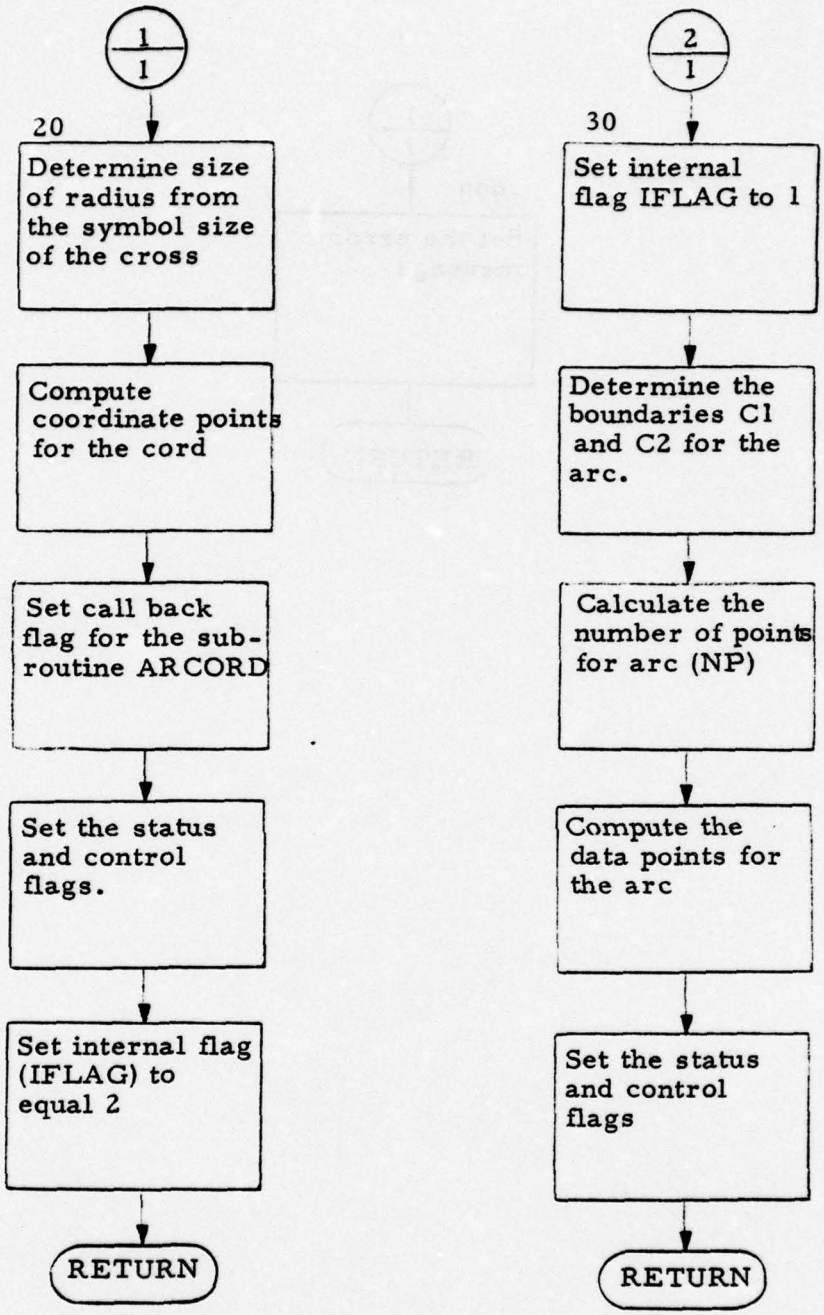



Figure III-41 - ARCORD Process Flow (Page 2 of 3)

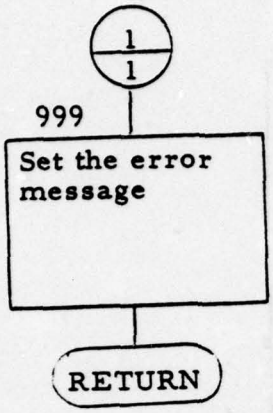



Figure III-41 - ARCORD Process Flow (Page 3 of 3)

OO. LINUP

1. Functional Description

LINUP Formats and prints out on the system line printer pertinent information about the job control parameters files processed and symbology generated.

2. Computer Definition

a. Core Memory

3054 octal words

b. Peripheral Equipment

System Line Printer

3. Program Description

a. Calling Routines

MONITR

b. Subroutines Used

None

c. Input

Common area (Process - Tally - Summary - Report)

d. Output

Line printer reports

e. Processing Methodology

LINUP is called when the symbolization processing is completed. Pertinent job control and processing information is maintained in common areas. LINUP simply retrieves the required information from the common area formats the data, and prints out the data on the system line printer.

f. Calling Sequence

CALL LINUP

g. Major Algorithms

None

4. Program Constants and Variables

Display format words

COUNT2 - tally of symbol pieces

5. Error Conditions

None

PP. SPEC

1. Functional Description

The function of the SPEC routine is to build/update symbol directive specification data files for a Graphic Line Symbolization System (GLSS) processing run. It is an independent routine of GLSS and is executed separately from other GLSS programs.

2. Computer Definition

a. Core Memory Used

2157 octal words

b. Peripheral Equipment

The peripheral equipment consists of a card reader and two permanent data files containing symbol specification data. File code (08) is a sequential file containing feature descriptor code data. File code (02) is a direct access file containing feature symbol piece directive information.

3. Program Description

a. Calling Routines

None.

b. Subroutines Used

None.

c. Input

Input to the SPEC routine consists of user supplied build/update symbol specification control and data cards. The control data card (card number 1) defines the activity to be executed (build new specification files or update existing files). The symbol specification data

cards consist of feature descriptor codes, color separation sheet numbers, symbol conformal - non-conformal information, symbol piece types, symbol piece sizes, and symbol piece line weights. See Volume I for the sequence and format of the data cards.

d. Output

The output from SPEC consists of two permanent data symbol specification files. The first file (file code 08) contains feature descriptor codes, stored on a permanent sequential data file. The second file (file code 02) consists of symbol piece specifications data stored on a permanent direct access data file. Figures III-43 and III-44 depict the two output files mentioned above.

e. Processing Methodology

The SPEC routine processing flow is depicted in Figure III-42. Upon entry the routine is initialized and the first user generated control card is read. In this card, the user will have specified the date (menmonic name IDATE1 and IDATE2) and one of the two functions of the SPEC (mnemonic name IMODE). The functions consist of one build mode (build new symbol specification files) or two, update mode (update existing symbol specification files). These above controls serve as a guide to the software for the reading, formatting and storing of the symbol specification data cards that follow. If the function is to build new specification files, a random logical record sequence number is calculated which is used for storage of the symbol specification for that feature. If the function is to update existing specification files, the random logical record sequence number to update is read from the next control card. The following processing, done in either mode, is to read symbol specification data cards containing feature descriptor codes (menmonic names ICODE1, ICODE2, and ICODE3), color separation sheet numbers (mnemonic names

ISTNO1 and ISTNO2), feature conformal information (mnemonic name ICON), symbol piece types (mnemonic name ITYPE), symbol piece size (mnemonic name SIZE), and symbol piece line weights (mnemonic name SYLWT). See Figure III-45 for the symbol piece type and numeric equivalence. They are then stored in their respective buffer areas with a line printer listing being generated. The above data cards are read until an end of symbol specification indication for that particular feature descriptor is detected. When this occurs, the random record sequence number input (mnemonic name IRWD), under update mode or generated under build mode (mnemonic name IR) is used as the random sequence storage pointer to write the symbol specifications to storage (permanent random file, file code 02). The overall process is repeated until an end of control card is detected (end of file on file code 05). When this occurs, the feature descriptor code buffer (mnemonic name IDESP) is output to a permanent sequential file (file code 08).

f. Calling Sequence

Not applicable

g. Major Algorithms

None

4. Program Constants and Variables

None

5. Error Conditions

None

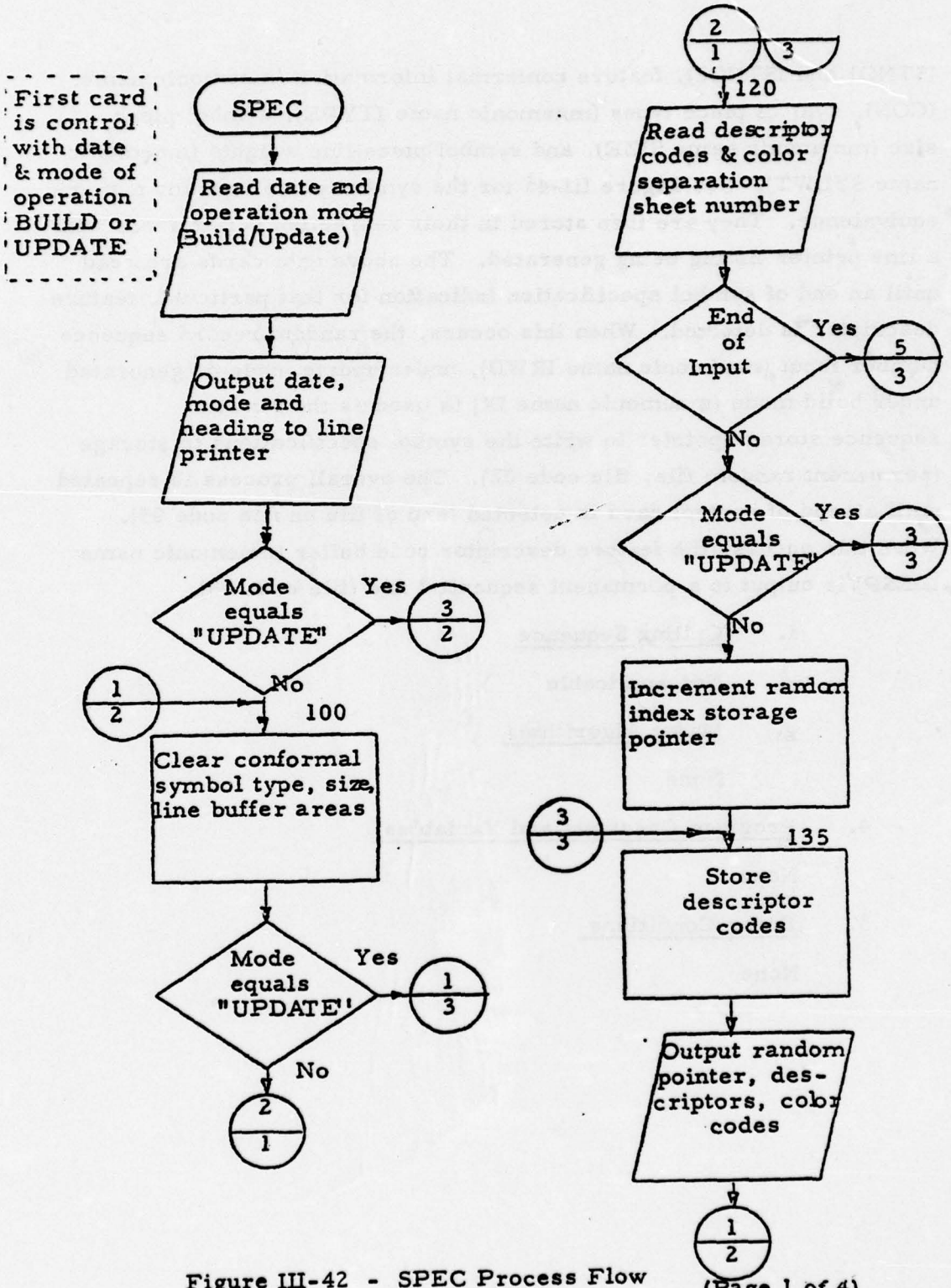



Figure III-42 - SPEC Process Flow (Page 1 of 4)

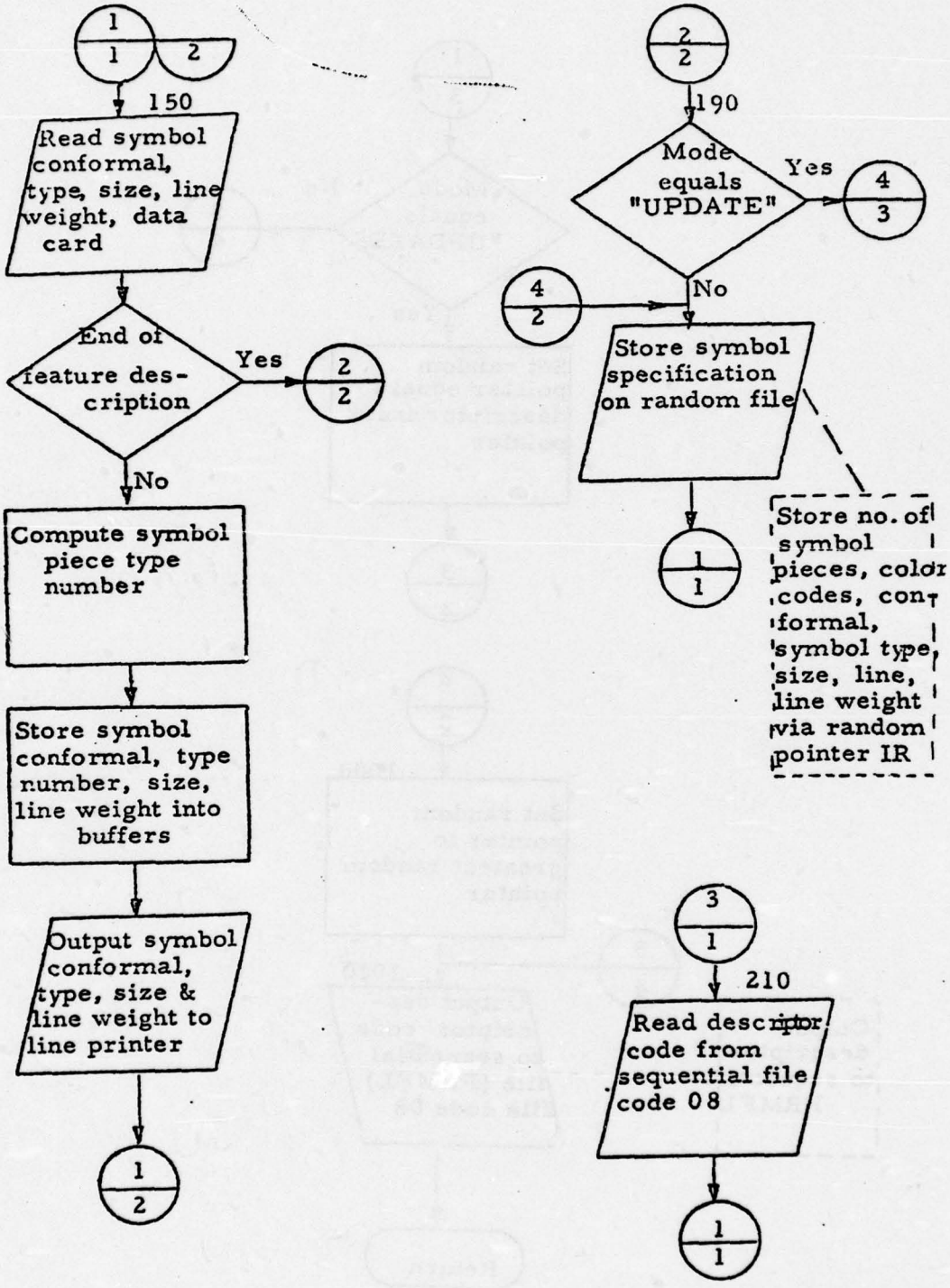



Figure III-42 - SPEC Process Flow (Page 2 of 4)

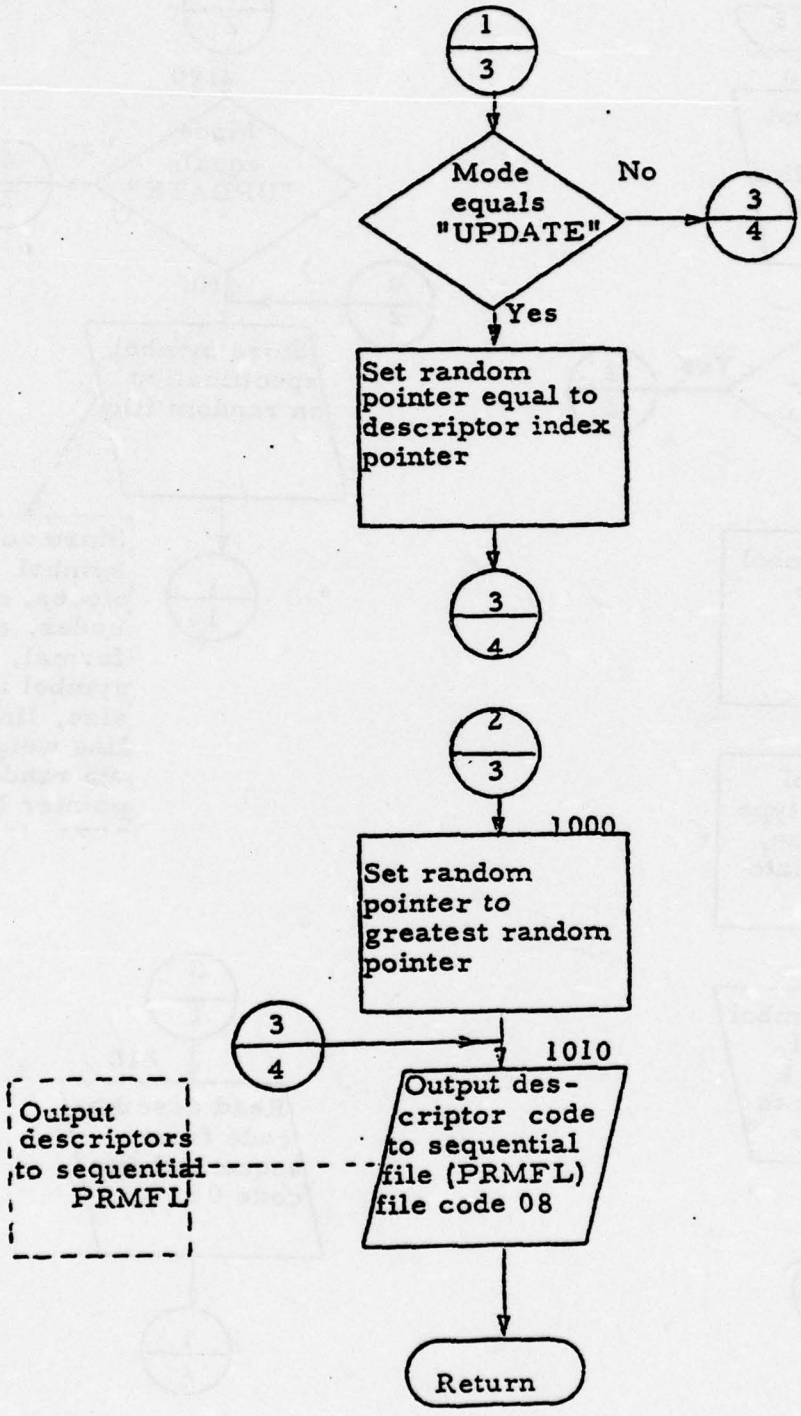



Figure III-42 - SPEC Process Flow

(Page 3 of 4)

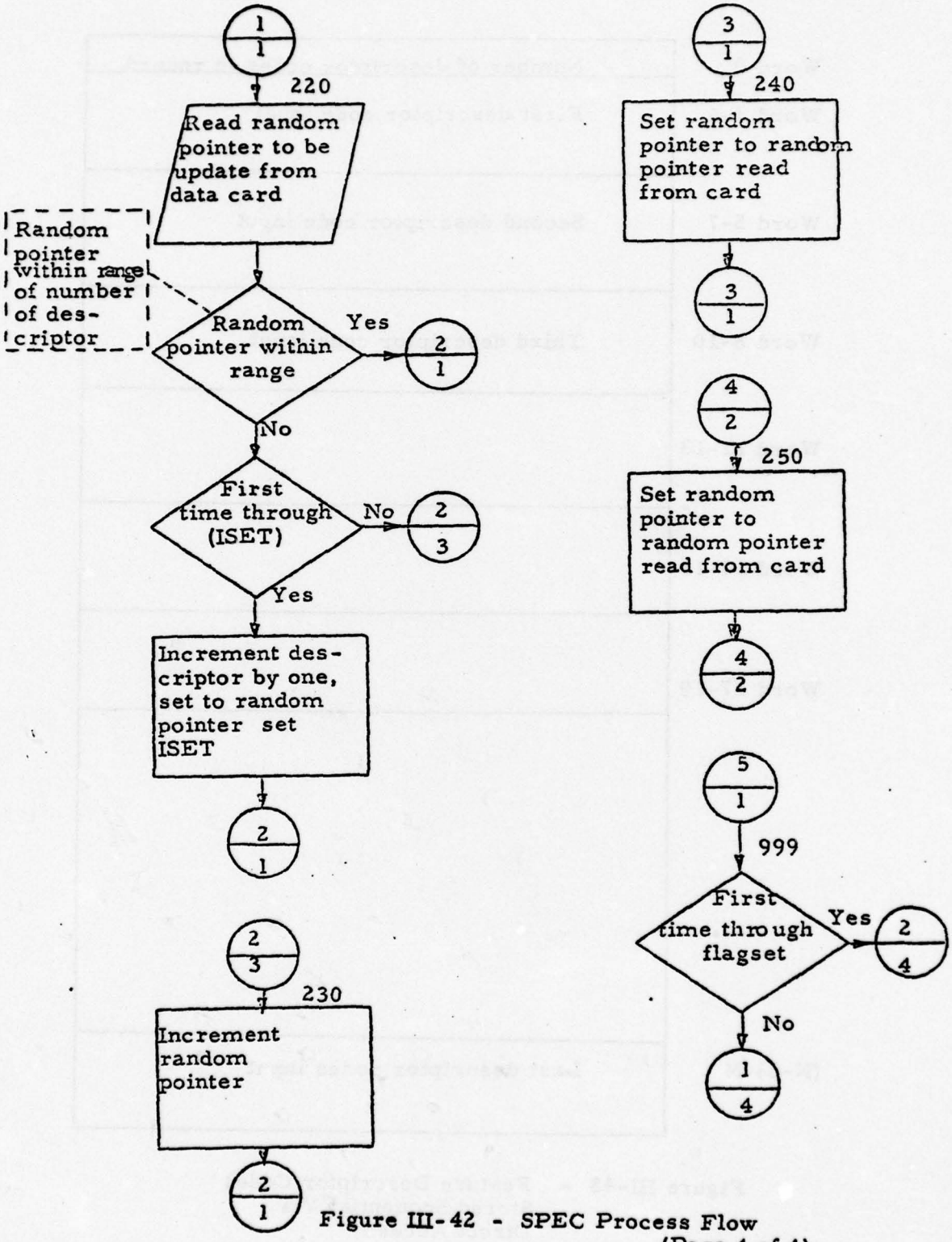



Figure III-42 - SPEC Process Flow
(Page 4 of 4)

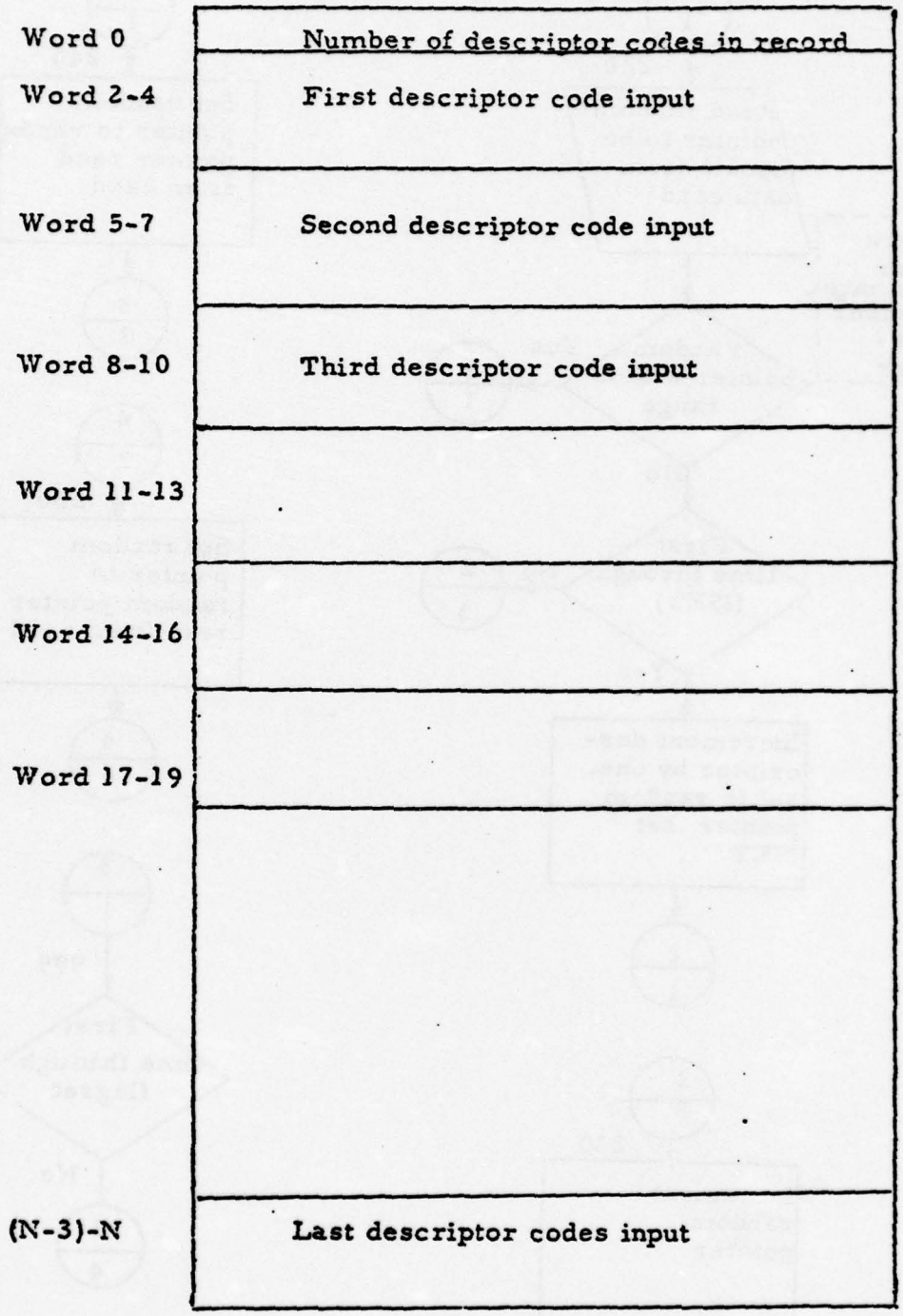



Figure III-43 - Feature Descriptor Codes Stored Sequential Via Direct Access

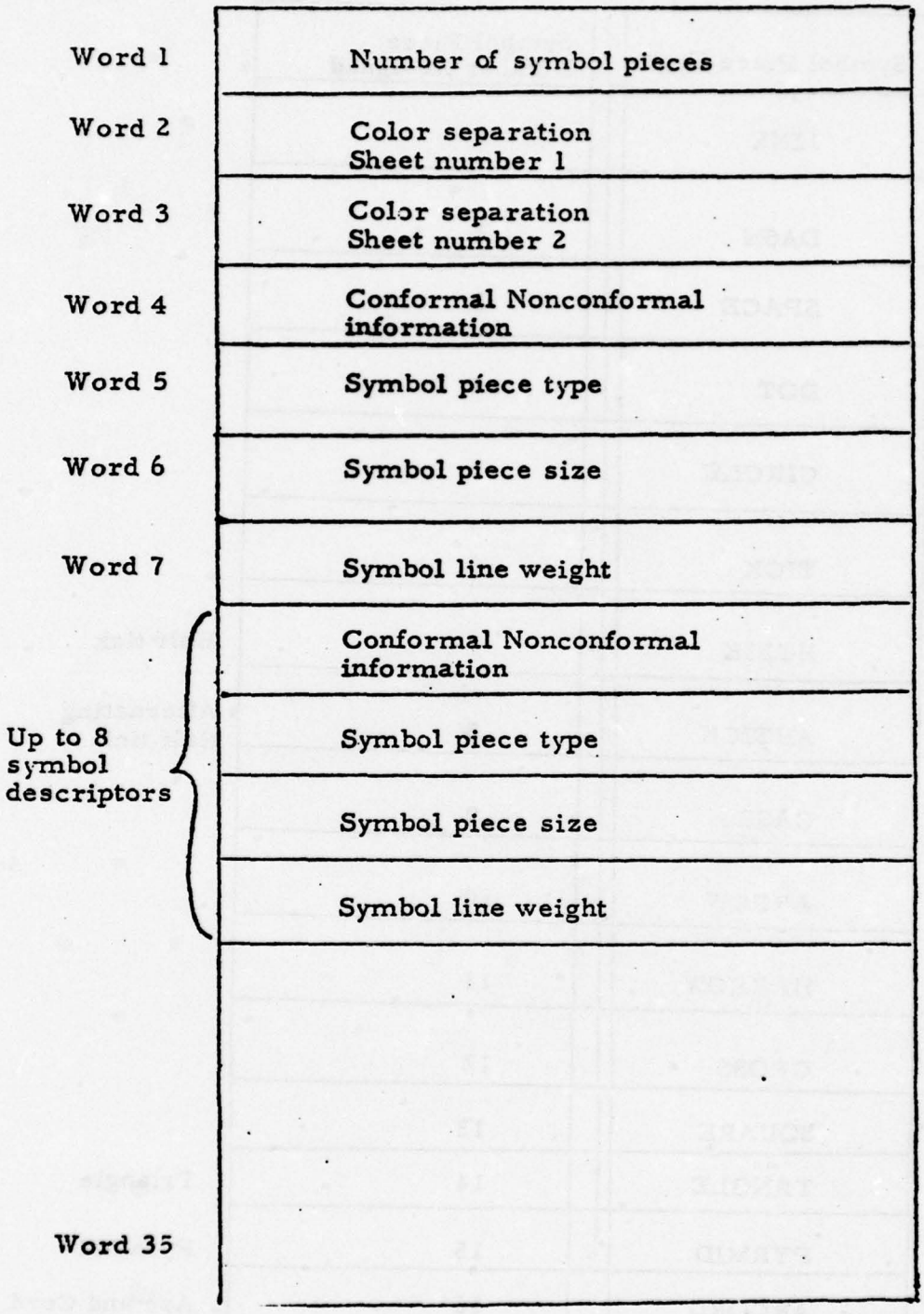



Figure III-44 - Symbol Piece Specification Record Stored Randomly Via Direct Access File

| Symbol Piece Type | Symbol Piece Number Assigned | |
|---|---|---|
| LINE | 1 | |
| DASH | 2 | |
| SPACE | 3 | |
| DOT | 4 | |
| CIRCLE | 5 | |
| TICK | 6 | |
| HTICK | 7 | Half tick |
| AHTICK | 8 | Alternating Half tick |
| CASE | 9 | |
| ARROW | 10 | |
| HARROW | 11 | |
| CROSS | 12 | |
| SQUARE | 13 | |
| TRNGLE | 14 | Triangle |
| PYRMID | 15 | Pyramid |
| ARCORD | 16 | Arc and Cord |

Figure III-45 - Symbol Piece Type Equivalance

QQ. HEADSUM

1. Functional Description

This routine uses REDREC and FORHED for reading and formating a LIS Table File. A printout is then generated which lists all headers and totals of their associated data points.

2. Computer Definition

a. Core Memory Used

121 octal words

b. Peripheral Equipment

Line printer

3. Program Description

a. Calling Routines

None

b. Subroutines Used

REDREC
FORHED

c. Input

LIS Table File

d. Output

Printout of file information.

e. Processing Methodology

HEADSUM calls REDREC for reading and inputing of all LIS records. FORHED is called to format headers prior printing out the feature class, type, subtype, descriptors and number of points representing the feature.

f. Calling Sequence

Not applicable.

g. Major Algorithms

None

4. Program Constants and Variables

MI - internal flag
NPTSI - tally of data points
NCNT - feature sequence counter

5. Error Conditions

None

# APPENDIX I

## COBOL AND FORTRAN COMMON AREAS

Copy available to DDC does not
permit fully legible reproduction

```
COMMON-STORAGE SECTION.
01  FEATURE-DESCRIPTOR-DATA COPY COBOLCOM.

    02  DUMMY PIC X(6) DISP-1.
    02  FEAT-CLASS-TYPE.
        03  F-CLASS PIC 99 DISP-1.
        03  F-TYPE  PIC 99 DISP-1.
        03  F-SUB-TYPE PIC 99 DISP-1.
        03  DESCRIPT1 PIC 9(6) DISP-1.
        03  DESCRIPT2 PIC 9(6) DISP-1.
    02  LEFT-RIGHT-CODE  PIC 9(10) COMP-4.
    02  FEAT-BOUND-REC.
        03  FEAT-X-MIN PIC 9(10) COMP-4.
        03  FEAT-Y-MIN PIC 9(10) COMP-4.
        03  FEAT-X-MAX PIC 9(10) COMP-4.
        03  FEAT-Y-MAX PIC 9(10) COMP-4.
    02  FEAT-FIRST-LAST.
        03  FEAT-X-FIRST PIC 9(10) COMP-4.
        03  FEAT-Y-FIRST PIC 9(10) COMP-4.
        03  FEAT-X-LAST  PIC 9(10) COMP-4.
        03  FEAT-Y-LAST  PIC 9(10) COMP-4.
    02  DESCRIPTOR-TEXT.
        03  TEXT1 PIC X(6) DISP-1.
        03  SP-FEAT-NO   PIC 9(10) COMP-4.
        03  SP-SYM-DIR   PIC 9(10) COMP-4 OCCURS 8 TIMES.
        03  TEXT2        PIC X(6) DISP-1 OCCURS 6 TIMES.
01  END-OF-FEAT-DEC PIC X(6) DISP-1 VALUE '111111'.
01  FEAT-LINE-CENTER-DATA.
    02  NUM-SPL-PTS PIC 9(10) COMP-4.
    02  SPL-DEX     PIC 9(10) COMP-4.
    02  SPECIAL-PTS PIC 9(10) COMP-4 OCCURS 20 TIMES.
    02  NUM-PTS-SUB.
        03  NUM-PTS-1 PIC 9(10) COMP-4.
        03  NUM-PTS-2 PIC 9(10) COMP-4.
        03  NUM-PTS-3 PIC 9(10) COMP-4.
        03  NUM-PTS-4 PIC 9(10) COMP-4.
        03  NUM-PTS-5 PIC 9(10) COMP-4.
    02  NUM-PTS REDEFINES NUM-PTS-SUB PIC 9(10) COMP-4
                OCCURS 5 TIMES.
    02  COR-DEX     PIC 9(10) COMP-4 OCCURS 5 TIMES.
    02  CUR-DEX     PIC 9(10) COMP-4.
    02  FT-DEX      PIC 9(10) COMP-4 OCCURS 5 TIMES.
    02  COOR-DATA-A.
        03  WORD-32 PIC X(192) DISP-1.
        03  FILLER PIC X(6) DISP-1 OCCURS 5968 TIMES.
    02  COOR-DATA-B REDEFINES COOR-DATA-A.
        03  COOR-DATA OCCURS 2000 TIMES.
            04  XVAL PIC 9(10) COMP-4.
            04  YVAL PIC 9(10) COMP-4.
            04  ZVAL PIC 9(10) COMP-4.
01  END-CENTER-DATA PIC X(6) DISP-1 VALUE '222222'.
01  SYMBOL-SPEC-DIRECTIVE.
    02  SYM-DEX          PIC 9(10) COMP-4.
    02  NUM-SYM-PIECES PIC 9(10) COMP-4.
    02  COLOR-SHEET-NU1 PIC 9(10) COMP-4.
    02  COLOR-SHEET-NU2  PIC 9(10) COMP-4.
    02  COR-NON-NORMAL   PIC 9(10) COMP-4 OCCURS 8 TIMES.
    02  SYM-TYPE     PIC 9(10) COMP-4 OCCURS 8 TIMES.
    02  SYM-SIZE     PIC 9(10) COMP-4 OCCURS 8 TIMES.
    02  SYM-LINE-WT PIC 9(10) COMP-4 OCCURS 8 TIMES.
01  END-SYMBOL-SPEC PIC X(6) DISP-1 VALUE '333333'.
01  SYMBOL-SPEC-LIB-OVERRIDE.
    02  VENDEX PIC 9(10) COMP-4.
    02  NUM-OVER    PIC 9(10) COMP-4.
    02  SYM-FCT OCCURS 10 TIMES.
        03  F-CLASS-OVER PIC 99 DISP-1.
        03  F-TYPE-OVER PIC 99 DISP-1.
        03  F-SUB-TYPE-OVER PIC 99 DISP-1.
```

```
    02 SYM-FC2 PIC X(6) DISP-1 OCCURS 10 TIMES.
    02 SYM-FC3 PIC X(6) DISP-1 OCCURS 10 TIMES.
    02 STAG1-OVER PIC 9(10) COMP-4 OCCURS 10 TIMES.
    02 STAG2-OVER PIC 9(10) COMP-4 OCCURS 10 TIMES.
    02 SYM-DIR-OVER1 OCCURS 10 TIMES.
       03 SYM-CON-NOR PIC 9(10) COMP-4 OCCURS 8 TIMES.
    02 SYM-DIR-OVER2 OCCURS 10 TIMES.
       03 SYM-TYPE-OVER PIC 9(10) COMP-4 OCCURS 8 TIMES.
    02 SYM-DIR-OVER3 OCCURS 10 TIMES.
       03 SYM-SZ-OVER PIC 9(10) COMP-4 OCCURS 8 TIMES.
    02 SYM-DIR-OVER4 OCCURS 10 TIMES.
       03 SYM-LW-OVER PIC 9(10) COMP-4 OCCURS 8 TIMES.
01  END-SPEC-OVERRIDE PIC X(6) DISP-1 VALUE '999999'.
01  STATUS-INDICATOR-FLAGS-PTRS.
    02 STORE-SPEC-HEAD PIC 9(10) COMP-4.
    02 READ-SPEC-HEAD PIC 9(10) COMP-4.
    02 SYM-PIECE-OVER PIC 9(10) COMP-4.
    02 READ-BUFF PIC 9(10) COMP-4.
    02 FEAT-CONT PIC 9(10) COMP-4.
    02 SYM-READY-OUT PIC 9(10) COMP-4.
    02 ERROR-ID PIC 9(10) COMP-4.
    02 SYMBOL-CALLBACK PIC 9(10) COMP-4.
    02 JELLY-NUM-OUT PIC 9(10) COMP-4 OCCURS 5 TIMES.
    02 NUM-PEC-CALLBACKS PIC 9(10) COMP-4.
    02 PEC-CALLBACKS PIC 9(10) COMP-4 OCCURS 5 TIMES.
    02 ABORT-JOB-ID PIC 9(10) COMP-4.
    02 JOB-THRU-FLAG PIC 9(10) COMP-4.
01  END-INDICATOR PIC X(6) DISP-1 VALUE '555555'.
01  PARAMETERS-VARIABLES.
    02 DATA-RES PIC 9(10) COMP-4.
    02 DIS-ALONG-SYM PIC 9(10) COMP-4.
    02 MAX-NUM-PTS PIC 9(10) COMP-4.
    02 IN-FILE PIC X(6) DISP-1.
    02 OUT-FILE PIC X(6) DISP-1.
    02 SMOOTH-TYPE PIC 9(10) COMP-4.
    02 MIN-DIST PIC 9(10) COMP-4.
    02 MAX-DIST PIC 9(10) COMP-4.
    02 SLOPE-DIST PIC 9(10) COMP-4.
    02 NEW-FEAT PIC 9(10) COMP-4.
01  END-PAR-VARIABLES PIC X(6) DISP-1 VALUE '000000'.
01  PROCESS-TALLY-SUMMARY-REPORT.
    02 TIME-DATE.
       03 MONTHS PIC 99 DISP-1.
       03 DAYS PIC 99 DISP-1.
       03 YEARS PIC 99 DISP-1.
       03 TIME-USED PIC 9(10) COMP-4.
    02 SYM-IO-SUM.
       03 HEADERS-IN PIC 9(10) COMP-4.
       03 DATA-RECS-IN PIC 9(10) COMP-4.
       03 HEADERS-OUT PIC 9(10) COMP-4.
       03 DATA-REC-OUT PIC 9(10) COMP-4.
       03 PTS-IN PIC 9(10) COMP-4.
       03 PTS-OUT PIC 9(10) COMP-4.
    02 SYMBOLS-GENERATED.
       03 LINE-CENTER PIC 9(10) COMP-4.
       03 LINE-PTS PIC 9(10) COMP-4.
       03 FEAT-DASHED PIC 9(10) COMP-4.
       03 DASH-PTS PIC 9(10) COMP-4.
       03 NUM-DASHS PIC 9(10) COMP-4.
       03 FEAT-CASED PIC 9(10) COMP-4.
       03 CASE-PTS PIC 9(10) COMP-4.
       03 NUM-CASED-ROADS PIC 9(10) COMP-4.
       03 FEAT-TICKED PIC 9(10) COMP-4.
       03 TICK-PTS PIC 9(10) COMP-4.
       03 NUM-TICKS PIC 9(10) COMP-4.
       03 NUM-HALF-TICKS PIC 9(10) COMP-4.
       03 NUM-DOTS PIC 9(10) COMP-4.
       03 NUM-CIRCLES PIC 9(10) COMP-4.
       03 NUM-ARROWS PIC 9(10) COMP-4.
       03 NUM-HALF-ARROWS PIC 9(10) COMP-4.
       03 NUM-CROSSES PIC 9(10) COMP-4.
    02 FILES-IN PIC 9(10) COMP-4.
    02 FILES-OUT PIC 9(10) COMP-4.
    02 SMOOTH-PROC.
       03 PTS-PROC PIC 9(10) COMP-4.
       03 PTS-DELETED PIC 9(10) COMP-4.
       03 PTS-PASSED PIC 9(10) COMP-4.
    02 ERR-MESS.
       03 NUM-MESS PIC 9(10) COMP-4.
       03 ERR-TEXT OCCURS 20 TIMES.
          04 MESSAGE PIC X(40) DISP-1.
01  END-SUMMARY-REPORT PIC X(6) DISP-1 VALUE '777777'.
01  LIS.
    02 FILLER PIC X(6) DISP-1 OCCURS 125 TIMES.
```

Copy available to DDC does not permit fully legible reproduction

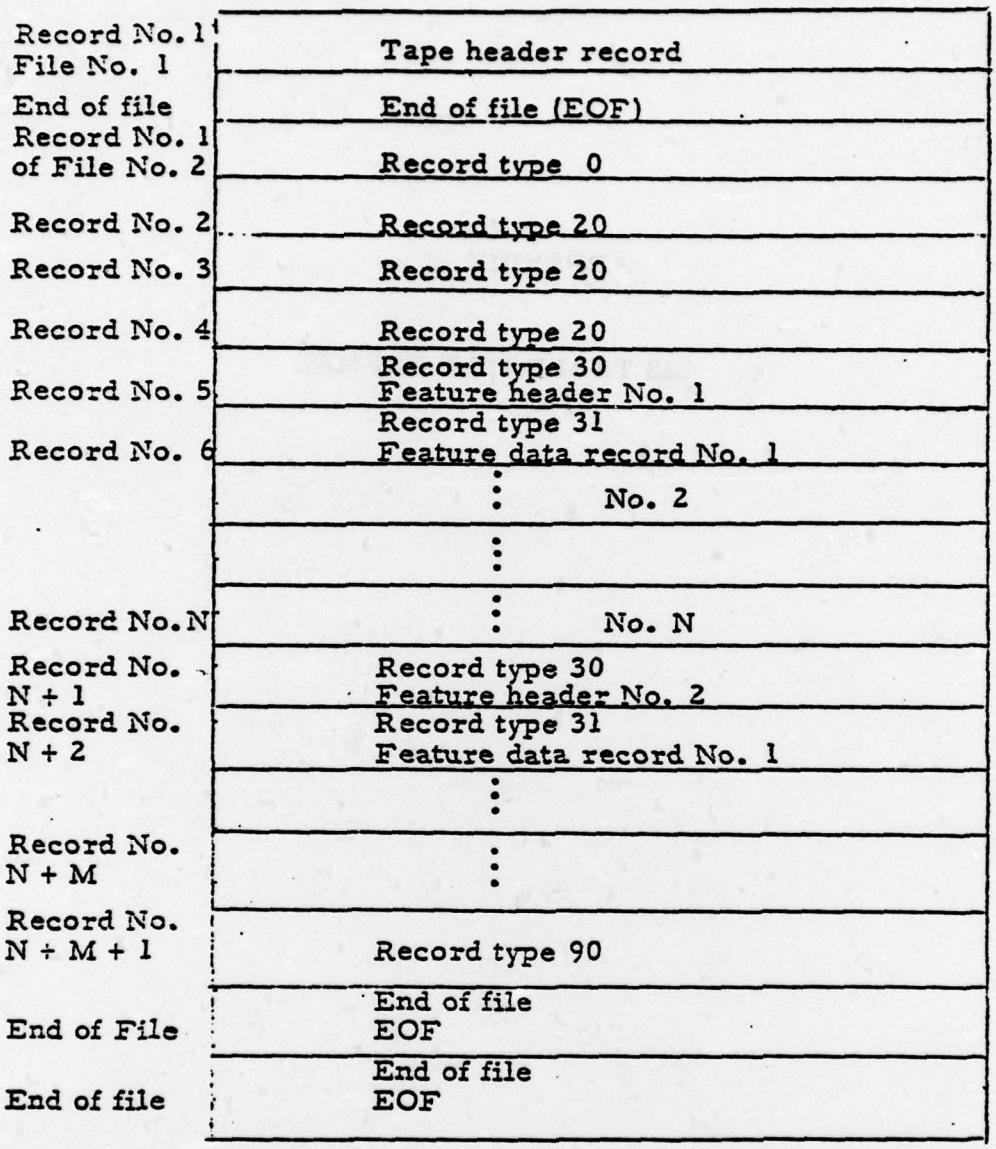
```
      COMMON IDUM,ICLSS1,ICLSS2,ICLSS3,LRCODE,IXMIN,IYMIN,IXMAX,
     * IYMAX,IXFST,IYFST,IXLST,IYLST,IHEAD(16),IC1,
C     ***************************************************
C     * ICLSS1 FEATURE CLASS,TYPE,SUB-TYPE
C     * ICLSS2 SIX CODIFIED DESCRIPTORS
C     * ICLSS3 TWO
C     * IXMIN    MINIMUM X
C     * IYMIN    MINIMUM Y
C     * IXMAX    MAXIMUM X
C     * IXMAX    MAXIMUM X
C     * IYMAX      '     Y
C     * IXFST    FIRST X
C     * IYFST      '     Y
C     * IXLST    LAST X
C     * IYLST      '     Y
C     * IHEAD(16) SIXTEEN TEXT WORDS
C     * IC1      CONTAINS BCD 1'S
C     ***************************************************
C
C
     * NSPTS,ISPDX,ISPPTS(20),NUMPTS(5),ICURDX(5),ICURDX,
     * IPTDX(5),IXYZ(3,2000),IC2,
C     ***************************************************
C     * FEATURE LINE CENTER DATA COMMON AREA
C     * NSPTS NUMBER OF SPECIAL POINTS
C     * ISPDX SPECIAL POINT INDEX
C     * ISPPTS(20) SPECIAL POINT BUFFER
C     * NUMPTS(5) NUMBER OF POINTS PER BUFFER
C     * ICURDX(5) STARTING INDEX POINTER INTO ONE
C     *        OF THE FIVE IXYZ BUFFERS
C     * ICURDX CURRENT BUFFER POINTER
C     * IPTDX(5) CURRENT INDEX POINTER
C     * IXYZ(3,2000) IXYZ(1,IPTDX(ICURDX)) X VALUE
C     *              '  2    '     '     Y   '
C     *              '  3    '     '     DISTANCE
C     * IC2  CONTAINS BCD 2'S
C     ***************************************************
C
C
C
     * ISTDEX,NUMPEC,ISTNO1,ISTNO2,ICONON(8),
     * ISYTP(8),ISYSZ(8),ISYPLW(8),IC3,
C     ***************************************************
C     * SYMBOL SPEC DIRECTIVES COMMON AREA
C     * ISTDEX SYMBOL SPEC DIRECTIVES INDEX
C     * NUMPEC NUMBER OF SYMBOL SPEC PIECES
C     * ISTNO1 COLOR SEPARATION SHEET NUMBER ONE
C     * ISTNO2 COLOR SEPARATION SHEET NUMBER TWO
C     * ICONON(8) CON-FORMAL NON-CON-FORMAL DATA
C     *      ZERO/CON-FORMAL,ONE/NON-CON-FORMAL
C     * ISYTP (8) SYMBOL PIECE TYPE
C     * ISYSZ (8) SYMBOL PIECE SIZE
C     * ISYPLW(8) SYMBOL PIECE LINE WEIGHT
C     * IC3      CONTAINS BCD 3'S
C     ***************************************************
C
C
C
     * IOVRDX,NUMOVR,IFCVR1(10),IFCVR2(10),
     * IFCVR3(10),ISTVR1(10),ISTVR2(10),ICNOCR(8,10),
     * ISTOVR(8,10),ISZOVR(8,10),ILWOVR(8,10),IC4,
C     ***************************************************
C     * SYMBOL SPEC DIRECTIVES OVERRIDE
C     * IOVRDX OVERRIDE INDEX
C     * NUMOVR NUMBER OF SPEC DIRECTIVE OVERRIDES
C     * IFCVR1(10) FEATURE CLASS,TYPE,SUB-TYPES
C     * IFCVR2(10) SIX BCD CODIFIED DESCRIPTORS
C     * IFCVR3(10) TWO    '     '     '
C     * ISTVR1(10) COLOR SEPARATION SHEET NUMBER ONE
C     * ISTVR2(10)   '     '     '     '     TWO
C     * ICNOCR(8,10) CON-FORMAL,NON-CON-FORMAL
C     * ISTOVR(8,10) SYMBOL TYPE
C     * ISZOVR(8,10) SYMBOL SIZE
C     * ILWOVR(8,10) SYMBOL LINE WEIGHT
C     * IC4      CONTAINS BCD 4'S
C     ***************************************************
C
C
C
```

```
     * IOTHED,IDIRCT,IOVRID,IRDBFF,IFTCNT,ISTRDY,IERRID,
     * ICLLBK,ITELRN(5),NUMCBK,IPECLK(5),IABORT,IJBEND,IC5,
C     ************************************************
C       * STATUS INDICATORS FLAGS AND POINTERS COMMON AREA
C       * IOTHED STORE SYMBOL SPEC IN HEADER FLAG
C       * IDIRCT READ SYMBOL SPEC FROM HEADER FLAG
C       * IOVRID OVERRIDE SYMBOL SPECIFICATION DIR
C       * IRDBFF READ BUFFER
C       * IFTCNT FEATURE CONTINUATION FLAG
C       * ISTRDY SYMBOL READY FOR OUTPUT FLAG
C       * IERRID
C       * ICLLBK SYMBOL SUBROUTINE CALL BACK FLAG SET
C       *      TO SYMBOL SUBROUTINE NUMBER
C       * ITELRN(5) TELLY RUN OUT FOR BUFFER IN QUESTION
C       * NUMCBK NUMBER OF CALL BACKS
C       * IPECLK(5) SYMBOL PIECE GENERATER CALL BACK FLAGS
C       * IABORT ABORT RUN FLAG
C       * IJBEND JOB END FLAG
C       * IC5 CONTAINS BCD 5'S
C     ************************************************
C
C
C
     * IDTRES,IDISYM,IMAXPT,IFILE,IOFILE,ISMOTH,
     * IMINDT,IMAXDT,ISPDST,NEWFET,IC6,
C     ************************************************
C       * PARAMETERS AND VARIABLE COMMON AREA
C       * IDTRES INPUT DATA RESOLUTION
C       * IDISYM DIATANCE ALONG SYMBOL PIECE
C       * IMAXPT MAXIMUM NUMBER OF POINTS TO INPUT
C       * IFILE  TYPE OF FORMAT OF INPUT FILE BCD
C       * IOFILE  '   '   '   '   OUTPUT FILE BCD
C       * ISMOTH SMOOTH OPTION
C       * IMINDT MINIMUM DISTANCE TO SMOOTH
C       * IMAXDT MAXIMUM DISTANCE FOR TRACE DATA
C       * ISPDST SLOPE DISTANCE
C       * NEWFET WHEN SET TO 1 NEW FEATURE
C       * IC6 CONTAINS BCD 6'S
C     ************************************************
C
C
C
     * IMDYR,ITMUSD,IHEDIN,IDTIN,IHEDOT,IDTOUT,NPTSIN,NPTSOT,
     * NLNCNT,NLNPTS,NFTDSH,NDSHPT,NUMDSH,
     * NFTCSD,NCSDPT,NCSDRD,NFTTKD,NTKDPT,NUMTKS,NUMHTK,NUMDOT,
     * NUMCIR,NUMARR,NUMHAR,NUMCRS,
     * IFILIN,IFILOT,IPTPRS,IPTDLT,IPTPSS,NUMERS,ITXERR(8,20),IC7,
C     ************************************************
C       * PROCESS TALLY SUMMARY REPORT COMMON AREA
C       * IMDYR MONTH DAY YEAR BCD
C       * ITMUSD PROCESSING TIME USED
C       * IHEDIN NUMBER OF HEADERS INPUT
C       * IDTIN DATA RECORDS INPUT
C       * IHEDOT NUMBER OF HEADERS OUTPUT
C       * IDTOUT DATA RESOLUTION OUTPUT
C       * NPTSIN NUMBER OF POINTS INPUT
C       * NPTSOT  '   '   '   OUTPUT
C       * NLNCT  NUMBER OF LINE CENTERS OUTPUT
C       * NLNPTS NUMBER OF POINTS OUTPPUT FOR LINE CENTER
C       * NFTDSH NUMBER OF FEATURES DASHED
C       * NDSHPT NUMBER OF POINTS OUTPUT FOR DASHES
C       * NUMDSH NUMBER OF DASHES
C       * NFTCSD NUMBER OF FEATURES CASED
C       * NCSDPT NUMBER OF CASED POINTS
C       * NCSDRD NUMBER OF CASED ROADS
C       * NFTTKD NUMBER OF FEATURS TICKED
C       * NTKDPT NUMBER OF POINTS OUTPUT FOR TICKS
C       * NUMTKS NUMBER OF TICKS GENERATED
C       * NUMHTK NUMBER OF HALF TICKS GENERATED
C       * NUMDOT NUMBER OF DOTS GENERATED
C       * NUMCIR NUMBER OF CIRCLES GENERATED
C       * NUMARR NUMBER OF ARROWS GENERATED
C       * NUMHAR NUMBER OF HALF ARROWS GENERATED
C       * NUMCRS NUMBER OF CROSS GENERATED
C       * IFILIN NUMBER OF FILES INPUT
C       * IFILOT NUMBER OF FILES OUTPUT
C       * IPTPRS NUMBER OF POINTS PROCESSED
C       * IPTDLT NUMBER OF POINTS DELETED
C       * IPTPSS NUMBER OF POINTS PASSED
```

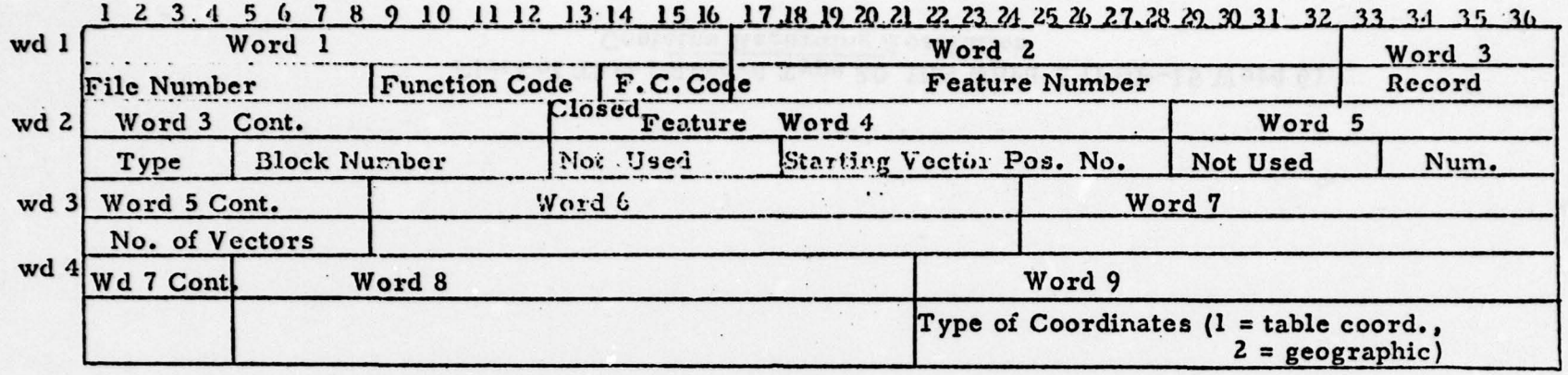
```
C         * NUMERS NUMBER OF SYMBOL GENERATION ERRORS
C         * ITAERR(6,20) BUFFER CONTAINING TEXT ERRORS
C         * IC7 CONTAINS BCD 7'S
C         *************************************************
C
C
C
C
C
C
      * IREC(1114),IRTYPE,IENDTP,IRSUIT,UNITS,
      *    IBLNUM,ISVPN,NUMVEC,IFETNO,IVECHD
C
C
C         *************************************************
C         * COMMON LIS MNEMONIC MEANINGS
C         * IREC   BUFFER CONTAINING LIS RECORD
C         * IRTYPE RECORD TYPE
C         * IENDTP END OF INPUT TAPE FLAG
C         * IRSUIT RECORDING RESOLUTION UNITS
C         * UNITS  UNITS OF MEASURE (UNITS=IRSUIT*INCHES)
C         * IBLNUM BLOCK NUMBER
C         * ISVPN  STARTING VECTOR POSITION NUMBER
C         * NUMVEC NUMBER OF VECTORS IN BLOCK
C         * IFETNO FEATURE NUMBER (FOUND IN LIS RECORD)
C         * IVECHD VECTORS IN HEADER FLAG
C         *
C         *
C         *
C         *************************************************
```

# APPENDIX II

## LIS TABLE FILE FORMAT

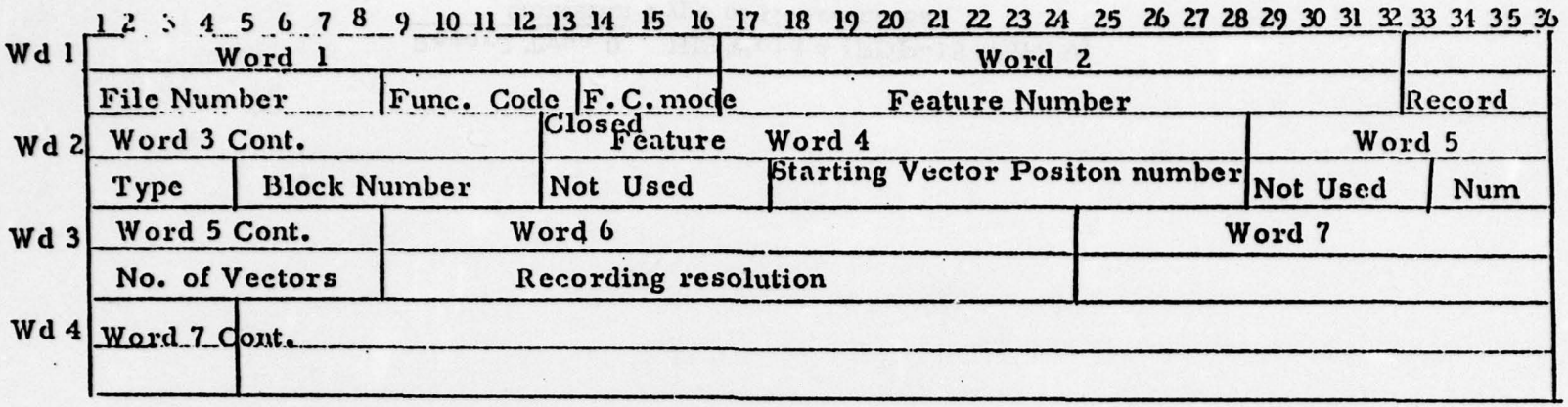



L.I.S. Magnetic Tape Layout



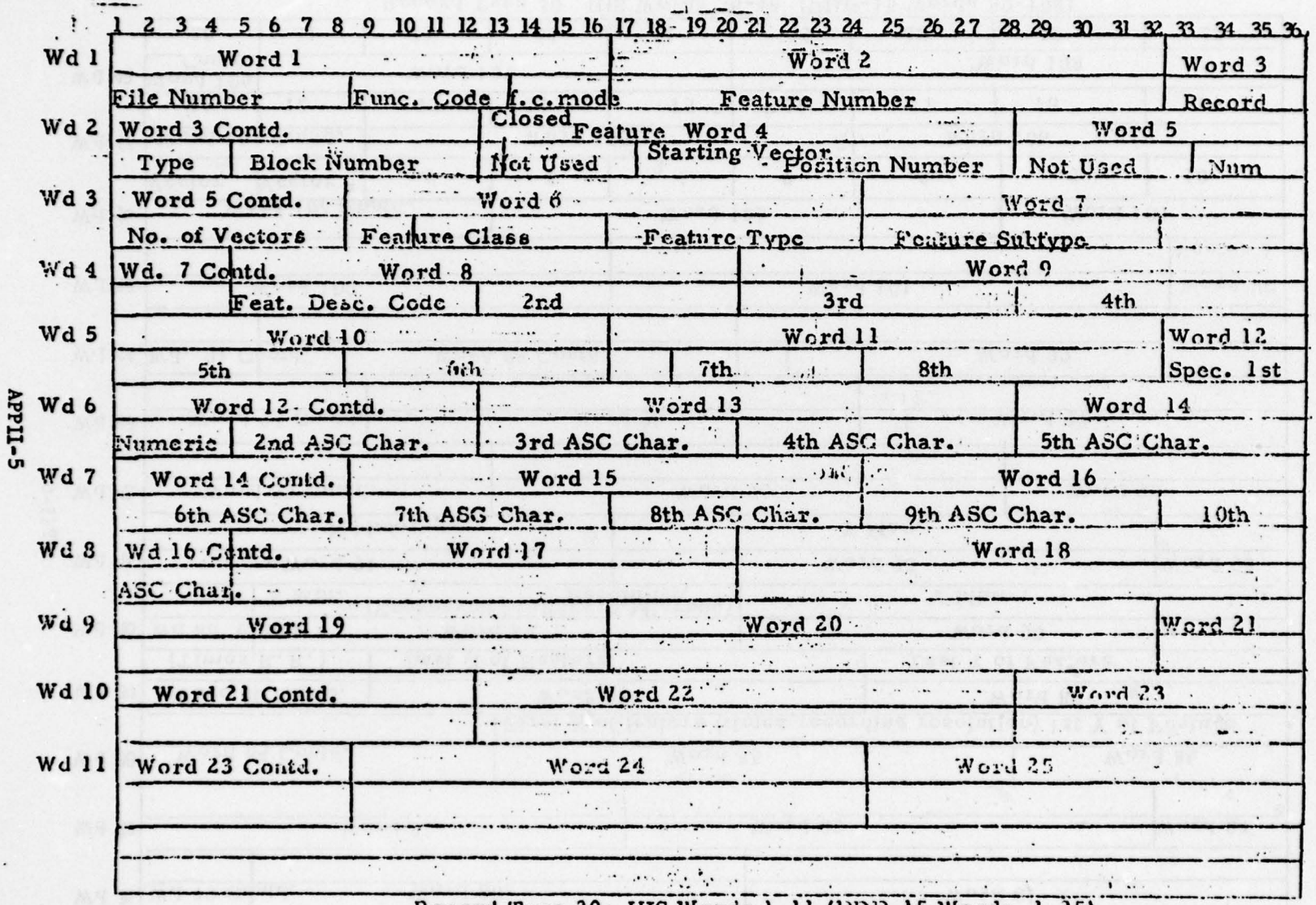



Record Type 0 HIS Word 4 (PDP-15 Word 9)
Contains Type of Coordinates

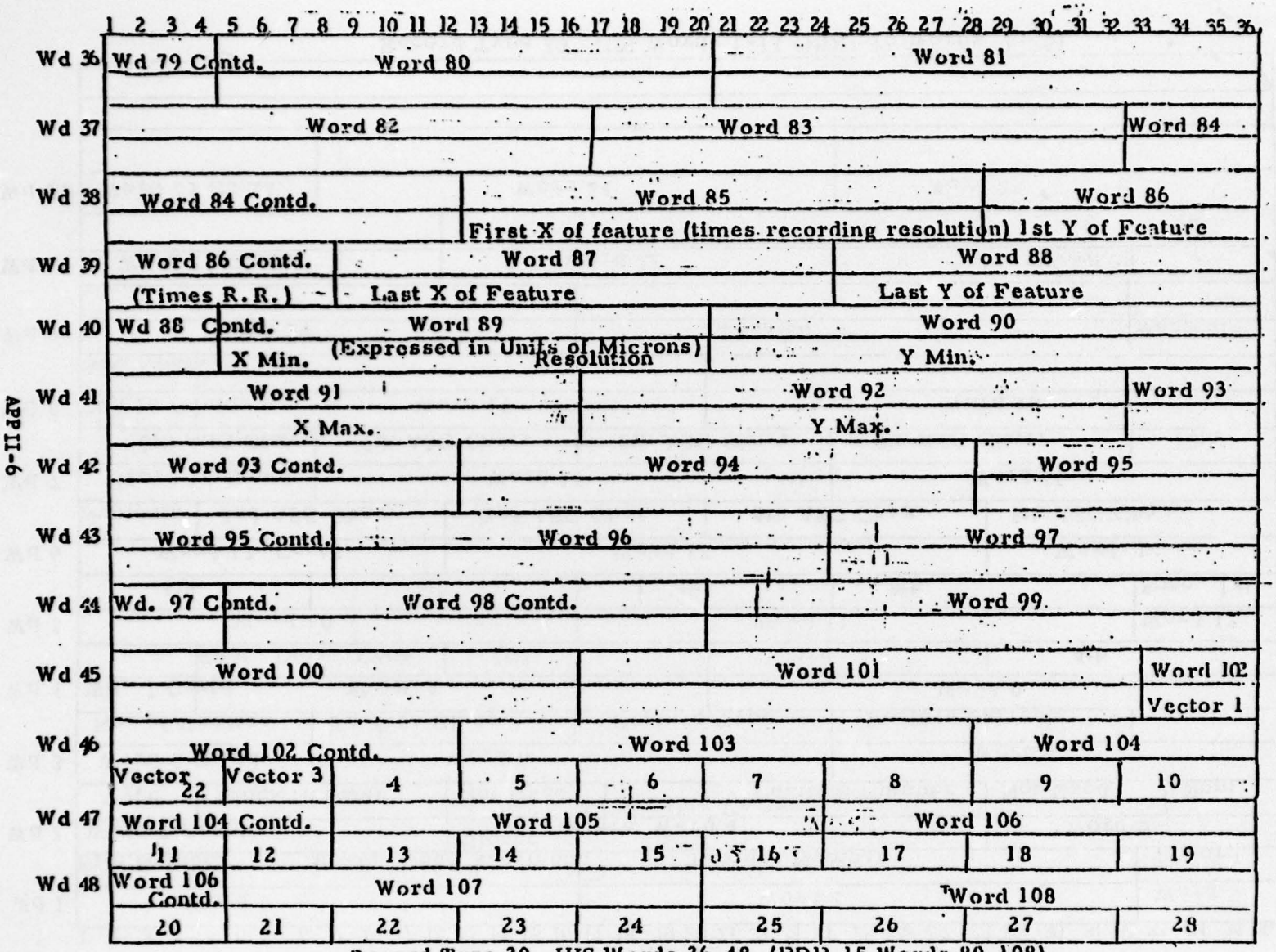





**First of Three Record Type 20 HIS Word 3 (PDP-15 Word 6)**
**Contains Recording Resolution**



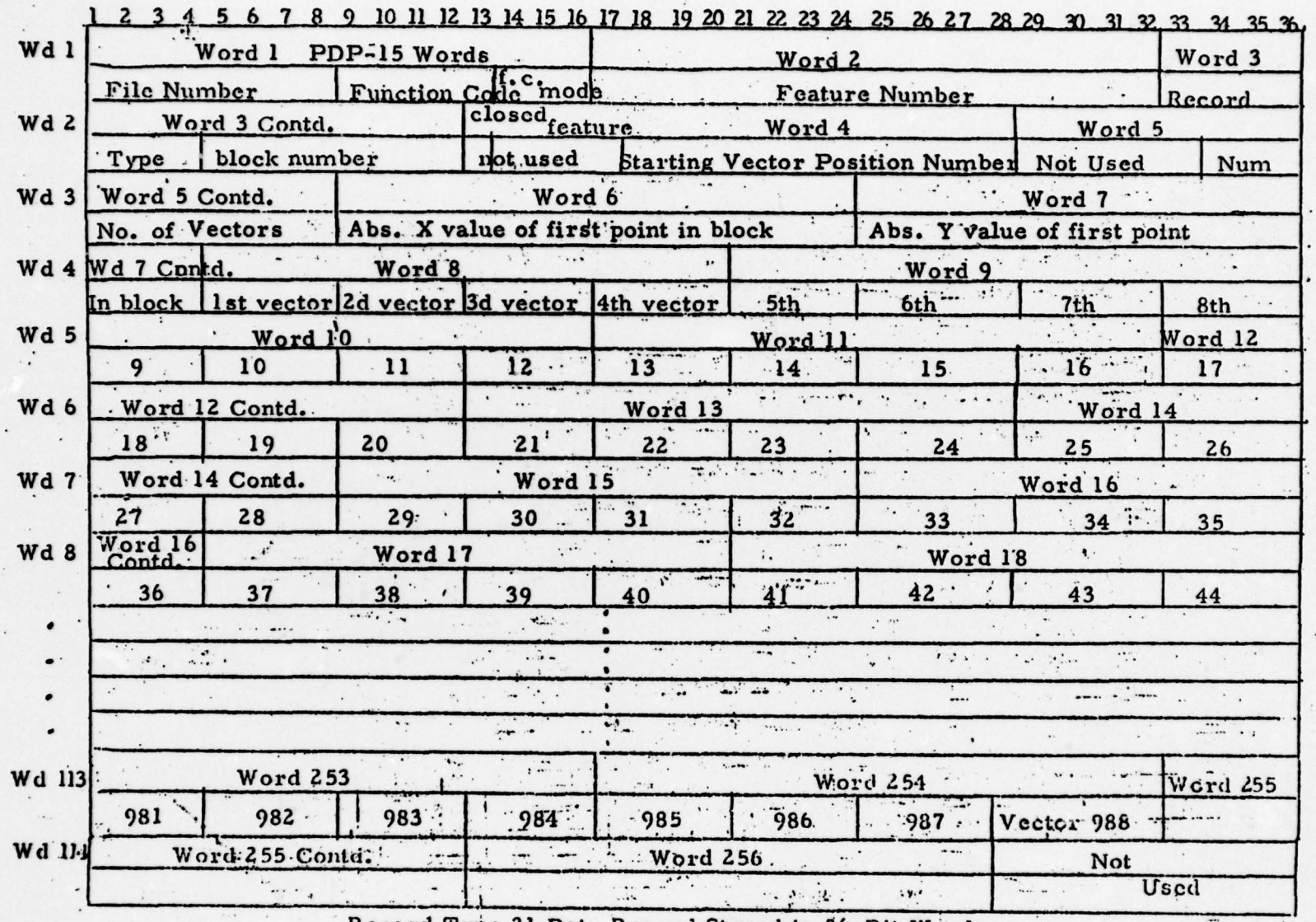

Record Type 30 HIS Words 1-11 (PDP-15 Words 1-25)




Record Type 30 HIS Words 36-48 (PDP-15 Words 80-108)

APII-6

61




Record Type 31 Data Record Stored in 36-Bit Words

# APPENDIX III

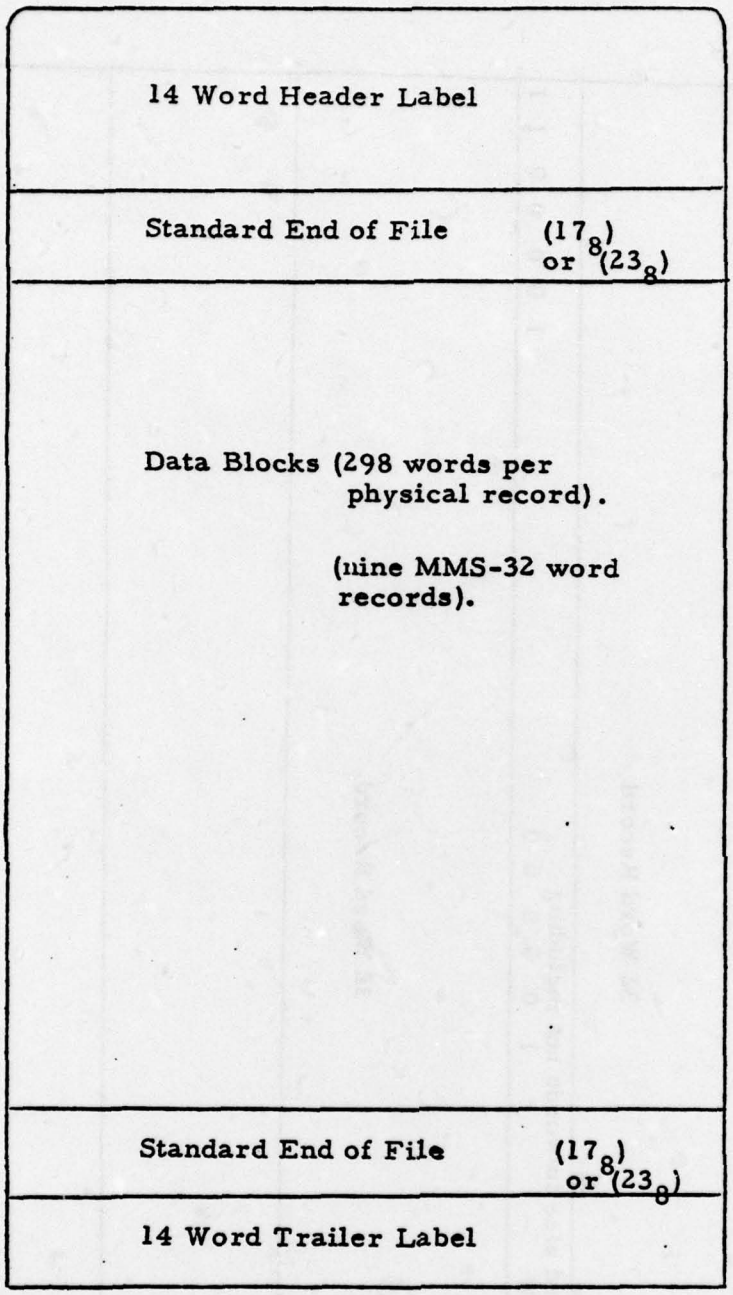
## MMS-32 WORD FILE FORMAT FOR DMAAC

HEADER RECORD FORMAT

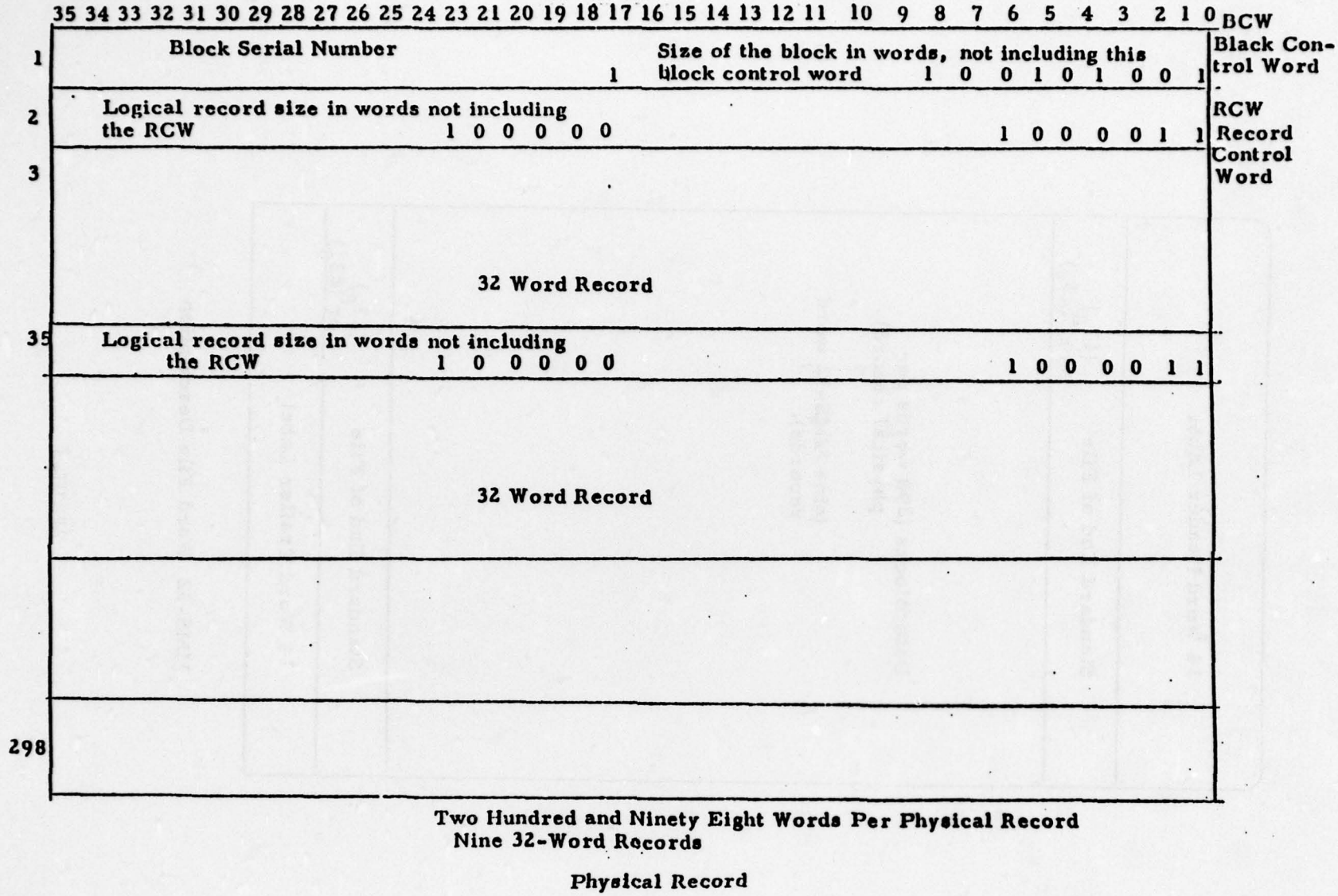
| Word | Contents |
|---|---|
| 1-4 | Zero |
| 5 | Record content, bits 5-0 (octal 34) |
| 6 | Feature code bits 11-0 (fieldata) |
| 7 | Symbol piece line weight bits 5-0 |
| 8 | Zero |
| 9 | Special Numerics (fieldata) |
| 10-24 | Zero |
| 25 | Minimum X value of bounding rectangle |
| 26 | Minimum Y value of bounding rectangle |
| 27 | Maximum X value of bounding rectangle |
| 28 | Maximum Y value of bounding rectangle |
| 29 | First X value of feature |
| 30 | First Y value of feature |
| 31 | Last X value of feature |
| 32 | Last Y value of feature |

DATA RECORD FORMAT

| Word | Contents |
|---|---|
| 1 | In bits 25-20 the number of meaningful words in record. |
| 2 | Absolute points in bits 35-21. |
| 3-32 | Data points of the feature (15 pairs of X, Y coordinates Honeywell 6000 floating point numbers). |




MMS-32 Word File Description




Two Hundred and Ninety Eight Words Per Physical Record
Nine 32-Word Records

Physical Record

# *MISSION*
# *of*
# *Rome Air Development Center*

*RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications ($C^3$) activities, and in the $C^3$ areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.*